CertiK Verification Report For Ocean Protocol

Request Date: 2019-04-03
Revision Date: 2019-04-08

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 Ocean Protocol(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 08, 2019

Summary

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

Type of Issues

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

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

Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0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

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Summary

The Ocean Token is implemented with good engineering quality, and strictly follows the standard ERC20 interface, with minimal set of additional features for central governance and life cycle managements. CertiK does not find any potential security risks with those add-on functionalities, however token holders should still be aware of the administrative authority of the contract owner, who is able to perform critical actions such as pause,

mint and kill. On the other hand, given the fact that the token is governed by the Ocean Protocol Foundation (OPF) via a Multisignature wallet, we believe the chance of token getting maliciously manipulated or tampered is low and ignorable. The multisig wallet smart contract is not within the service scope of this audit, thus we cannot provide any assessment or recommendations.

The additional features on top of the standard token mostly focus on data storage and access invokable by token owner. Basically, an array of wallet addresses who hold non-zero balance of Ocean Token is stored in smart contract and will be accessed by owner later for the purpose of migrating the balance from the erc20 token to its mainnet token. The token itself is non-payable (we assume the payable logic is handled on the multisig wallet side) and there is fallback function implemented to revert any value sent, which greatly mitigated the potential risk of being hacked.

The contract leans on appropriate standards with minimal storage to fulfill the business requirements and proper intervention mechanism to prevent human errors. We conclude that Ocean Token shall launch in a well-tested and secure state, is not vulnerable to any known antipatterns or bugs, and the risk is likely very low.

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 OceanToken.sol

```
1  pragma solidity 0.5.3;
2
3  import 'openzeppelin-solidity/contracts/token/ERC20/ERC20Capped.sol';
4  import 'openzeppelin-solidity/contracts/token/ERC20/ERC20Detailed.sol';
5  import 'openzeppelin-solidity/contracts/token/ERC20/ERC20Pausable.sol';
6  import 'openzeppelin-solidity/contracts/ownership/Ownable.sol';
7
8  /**
9   * @title Ocean Protocol ERC20 Token Contract
10  * @author Ocean Protocol Team
11  * @dev Implementation of the Ocean Token.
12  */
13 contract OceanToken is Ownable, ERC20Pausable, ERC20Detailed, ERC20Capped {
14
15     using SafeMath for uint256;
16
17     uint8 constant DECIMALS = 18;
18     uint256 constant CAP = 1410000000;
19     uint256 TOTALSUPPLY = CAP.mul(uint256(10) ** DECIMALS);
20
21     // keep track token holders
22     address[] private accounts = new address[](0);
23     mapping(address => bool) private tokenHolders;
24
25     /**
26      * @dev OceanToken constructor
27      * @param contractOwner refers to the owner of the contract
28      */
29     constructor(
30         address contractOwner
31     )
32         public
33         ERC20Detailed('OceanToken', 'OCEAN', DECIMALS)
34         ERC20Capped(TOTALSUPPLY)
35         Ownable()
36     {
37         addPauser(contractOwner);
38         renouncePauser();
39         addMinter(contractOwner);
40         renounceMinter();
41         transferOwnership(contractOwner);
42     }
43
44     /**
45      * @dev transfer tokens when not paused (pausable transfer function)
46      * @param _to receiver address
47      * @param _value amount of tokens
48      * @return true if receiver is illegible to receive tokens
49      */
50     /*@CTK _transfer
51       @tag assume_completion
52       @pre msg.sender != _to
53       @post _to != address(0)
54       @post __post._balances[msg.sender] == _balances[msg.sender] - _value
```

```
55      @post __post._balances[_to] == _balances[_to] + _value
56     */
57    function transfer(
58        address _to,
59        uint256 _value
60    )
61        public
62        returns (bool)
63    {
64        bool success = super.transfer(_to, _value);
65        if (success) {
66            updateTokenHolders(msg.sender, _to);
67        }
68        return success;
69    }
70
71    /**
72     * @dev transferFrom transfers tokens only when token is not paused
73     * @param _from sender address
74     * @param _to receiver address
75     * @param _value amount of tokens
76     * @return true if receiver is illegible to receive tokens
77     */
78    /*@CTK transfer_from
79      @tag assume_completion
80      @pre _from != _to
81      @post _to != address(0)
82      @post _value <= _allowed[_from][msg.sender]
83      @post __post._balances[_from] == _balances[_from] - _value
84      @post __post._balances[_to] == _balances[_to] + _value
85      @post __post._allowed[_from][msg.sender] ==
86      _allowed[_from][msg.sender] - _value
87     */
88    function transferFrom(
89        address _from,
90        address _to,
91        uint256 _value
92    )
93        public
94        returns (bool)
95    {
96        bool success = super.transferFrom(_from, _to, _value);
97        if (success) {
98            updateTokenHolders(_from, _to);
99        }
100       return success;
101   }
102
103   /**
104    * @dev retrieve the address & token balance of token holders (each time retrieve
           partial from the list)
105    * @param _start index
106    * @param _end index
107    * @return array of accounts and array of balances
108    */
109   function getAccounts(
110       uint256 _start,
111       uint256 _end
```

```
112     )
113         external
114         view
115         onlyOwner
116         returns (address[] memory, uint256[] memory)
117     {
118         require(
119             _start <= _end && _end < accounts.length,
120             'Array index out of bounds'
121         );
122 
123         uint256 length = _end.sub(_start).add(1);
124 
125         address[] memory _tokenHolders = new address[](length);
126         uint256[] memory _tokenBalances = new uint256[](length);
127 
128         for (uint256 i = _start; i <= _end; i++)
129         {
130             address account = accounts[i];
131             uint256 accountBalance = super.balanceOf(account);
132             if (accountBalance > 0)
133             {
134                 _tokenBalances[i] = accountBalance;
135                 _tokenHolders[i] = account;
136             }
137         }
138 
139         return (_tokenHolders, _tokenBalances);
140     }
141 
142     /**
143      * @dev get length of account list
144      */
145     /*@CTK getAccountsLength
146       @tag assume_completion
147       @post _owner == msg.sender
148       @post __return == accounts.length
149      */
150     function getAccountsLength()
151         external
152         view
153         onlyOwner
154         returns (uint256)
155     {
156         return accounts.length;
157     }
158 
159     /**
160      * @dev kill the contract and destroy all tokens
161      */
162     function kill()
163         external
164         onlyOwner
165     {
166         selfdestruct(address(uint160(owner())));
167     }
168 
169     /**
```

```
170      * @dev fallback function prevents ether transfer to this contract
171      */
172     function()
173         external
174         payable
175     {
176         revert('Invalid ether transfer');
177     }
178
179     /*
180      * @dev tryToAddTokenHolder try to add the account to the token holders structure
181      * @param account address
182      */
183     /*@CTK tryToAddTokenHolder
184       @tag assume_completion
185       @pre !tokenHolders[account] && _balances[account] > 0
186       @post __post.accounts[accounts.length] == account
187       @post __post.tokenHolders[account]
188      */
189     function tryToAddTokenHolder(
190         address account
191     )
192         private
193     {
194         if (!tokenHolders[account] && super.balanceOf(account) > 0)
195         {
196             accounts.push(account);
197             tokenHolders[account] = true;
198         }
199     }
200
201     /*
202      * @dev updateTokenHolders maintains the accounts array and set the address as a
            promising token holder
203      * @param sender address
204      * @param receiver address.
205      */
206     /*@CTK updateTokenHolders
207       @tag assume_completion
208       @pre sender != receiver
209       @pre !tokenHolders[sender] && _balances[sender] > 0
210       @pre !tokenHolders[receiver] && _balances[receiver] > 0
211       @post __post.accounts[accounts.length] == sender
212       @post __post.tokenHolders[sender]
213       @post __post.accounts[accounts.length + 1] == receiver
214       @post __post.tokenHolders[receiver]
215      */
216     function updateTokenHolders(
217         address sender,
218         address receiver
219     )
220         private
221     {
222         tryToAddTokenHolder(sender);
223         tryToAddTokenHolder(receiver);
224     }
225 }
```

File Migrations.sol

```
pragma solidity >=0.4.21 <0.6.0;

contract Migrations {
    address public owner;
    uint public last_completed_migration;

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

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

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

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

File openzeppelin-solidity/contracts/token/ERC20/ERC20.sol

```
pragma solidity ^0.5.0;

import "./IERC20.sol";
import "../../math/SafeMath.sol";

/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
 * Originally based on code by FirstBlood:
 * https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.sol
 *
 * This implementation emits additional Approval events, allowing applications to reconstruct the allowance status for
 * all accounts just by listening to said events. Note that this isn't required by the specification, and other
 * compliant implementations may not do it.
 */
contract ERC20 is IERC20 {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

```

```
23      mapping (address => mapping (address => uint256)) private _allowed;
24
25      uint256 private _totalSupply;
26
27      /**
28      * @dev Total number of tokens in existence
29      */
30      /*@CTK totalSupply
31        @post __return == _totalSupply
32       */
33      function totalSupply() public view returns (uint256) {
34          return _totalSupply;
35      }
36
37      /**
38      * @dev Gets the balance of the specified address.
39      * @param owner The address to query the balance of.
40      * @return An uint256 representing the amount owned by the passed address.
41      */
42      /*@CTK balanceOf
43        @post __return == _balances[owner]
44       */
45      function balanceOf(address owner) public view returns (uint256) {
46          return _balances[owner];
47      }
48
49      /**
50       * @dev Function to check the amount of tokens that an owner allowed to a spender.
51       * @param owner address The address which owns the funds.
52       * @param spender address The address which will spend the funds.
53       * @return A uint256 specifying the amount of tokens still available for the
             spender.
54       */
55      /*@CTK allowance
56        @post __return == _allowed[owner][spender]
57       */
58      function allowance(address owner, address spender) public view returns (uint256) {
59          return _allowed[owner][spender];
60      }
61
62      /**
63      * @dev Transfer token for a specified address
64      * @param to The address to transfer to.
65      * @param value The amount to be transferred.
66      */
67      /*@CTK transfer
68        @tag assume_completion
69        @pre msg.sender != to
70        @post to != address(0)
71        @post value <= _balances[msg.sender]
72        @post __post._balances[to] == _balances[to] + value
73        @post __post._balances[msg.sender] == _balances[msg.sender] - value
74       */
75      function transfer(address to, uint256 value) public returns (bool) {
76          _transfer(msg.sender, to, value);
77          return true;
78      }
79
```

```
80      /**
81       * @dev Approve the passed address to spend the specified amount of tokens on
             behalf of msg.sender.
82       * Beware that changing an allowance with this method brings the risk that someone
              may use both the old
83       * and the new allowance by unfortunate transaction ordering. One possible
             solution to mitigate this
84       * race condition is to first reduce the spender's allowance to 0 and set the
             desired value afterwards:
85       * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
86       * @param spender The address which will spend the funds.
87       * @param value The amount of tokens to be spent.
88       */
89      /*@CTK approve
90        @tag assume_completion
91        @post spender != address(0)
92        @post __post._allowed[msg.sender][spender] == value
93       */
94      function approve(address spender, uint256 value) public returns (bool) {
95          require(spender != address(0));
96
97          _allowed[msg.sender][spender] = value;
98          emit Approval(msg.sender, spender, value);
99          return true;
100     }
101
102     /**
103      * @dev Transfer tokens from one address to another.
104      * Note that while this function emits an Approval event, this is not required as
             per the specification,
105      * and other compliant implementations may not emit the event.
106      * @param from address The address which you want to send tokens from
107      * @param to address The address which you want to transfer to
108      * @param value uint256 the amount of tokens to be transferred
109      */
110     /*@CTK transfer_from
111       @tag assume_completion
112       @pre from != to
113       @post to != address(0)
114       @post value <= _allowed[from][msg.sender]
115       @post __post._balances[from] == _balances[from] - value
116       @post __post._balances[to] == _balances[to] + value
117       @post __post._allowed[from][msg.sender] ==
118       _allowed[from][msg.sender] - value
119      */
120     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
121         _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
122         _transfer(from, to, value);
123         emit Approval(from, msg.sender, _allowed[from][msg.sender]);
124         return true;
125     }
126
127     /**
128      * @dev Increase the amount of tokens that an owner allowed to a spender.
129      * approve should be called when allowed_[_spender] == 0. To increment
130      * allowed value is better to use this function to avoid 2 calls (and wait until
131      * the first transaction is mined)
```

```
132      * From MonolithDAO Token.sol
133      * Emits an Approval event.
134      * @param spender The address which will spend the funds.
135      * @param addedValue The amount of tokens to increase the allowance by.
136      */
137     /*@CTK increaseAllowance
138       @tag assume_completion
139       @post spender != address(0)
140       @post __post._allowed[msg.sender][spender] ==
141           _allowed[msg.sender][spender] + addedValue
142      */
143     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
144         require(spender != address(0));
145 
146         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].add(addedValue);
147         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
148         return true;
149     }
150 
151     /**
152      * @dev Decrease the amount of tokens that an owner allowed to a spender.
153      * approve should be called when allowed_[_spender] == 0. To decrement
154      * allowed value is better to use this function to avoid 2 calls (and wait until
155      * the first transaction is mined)
156      * From MonolithDAO Token.sol
157      * Emits an Approval event.
158      * @param spender The address which will spend the funds.
159      * @param subtractedValue The amount of tokens to decrease the allowance by.
160      */
161     /*@CTK decreaseAllowance
162       @tag assume_completion
163       @post spender != address(0)
164       @post __post._allowed[msg.sender][spender] ==
165           _allowed[msg.sender][spender] - subtractedValue
166      */
167     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
168         require(spender != address(0));
169 
170         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].sub(
                subtractedValue);
171         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
172         return true;
173     }
174 
175     /**
176     * @dev Transfer token for a specified addresses
177     * @param from The address to transfer from.
178     * @param to The address to transfer to.
179     * @param value The amount to be transferred.
180     */
181     /*@CTK _transfer
182       @tag assume_completion
183       @pre from != to
184       @post to != address(0)
185       @post __post._balances[from] == _balances[from] - value
186       @post __post._balances[to] == _balances[to] + value
```

```
187      */
188     function _transfer(address from, address to, uint256 value) internal {
189         require(to != address(0));
190
191         _balances[from] = _balances[from].sub(value);
192         _balances[to] = _balances[to].add(value);
193         emit Transfer(from, to, value);
194     }
195
196     /**
197      * @dev Internal function that mints an amount of the token and assigns it to
198      * an account. This encapsulates the modification of balances such that the
199      * proper events are emitted.
200      * @param account The account that will receive the created tokens.
201      * @param value The amount that will be created.
202      */
203     /*@CTK _mint
204       @tag assume_completion
205       @post account != 0
206       @post __post._totalSupply == _totalSupply + value
207       @post __post._balances[account] == _balances[account] + value
208      */
209     function _mint(address account, uint256 value) internal {
210         require(account != address(0));
211
212         _totalSupply = _totalSupply.add(value);
213         _balances[account] = _balances[account].add(value);
214         emit Transfer(address(0), account, value);
215     }
216
217     /**
218      * @dev Internal function that burns an amount of the token of a given
219      * account.
220      * @param account The account whose tokens will be burnt.
221      * @param value The amount that will be burnt.
222      */
223     /*@CTK _burn
224       @tag assume_completion
225       @post account != 0
226       @post value <= _balances[account]
227       @post __post._totalSupply == _totalSupply - value
228       @post __post._balances[account] == _balances[account] - value
229      */
230     function _burn(address account, uint256 value) internal {
231         require(account != address(0));
232
233         _totalSupply = _totalSupply.sub(value);
234         _balances[account] = _balances[account].sub(value);
235         emit Transfer(account, address(0), value);
236     }
237
238     /**
239      * @dev Internal function that burns an amount of the token of a given
240      * account, deducting from the sender's allowance for said account. Uses the
241      * internal burn function.
242      * Emits an Approval event (reflecting the reduced allowance).
243      * @param account The account whose tokens will be burnt.
244      * @param value The amount that will be burnt.
```

```
245      */
246     /*@CTK _burnFrom
247       @tag assume_completion
248       @post value <= _allowed[account][msg.sender]
249       @post __post._allowed[account][msg.sender] == _allowed[account][msg.sender] -
              value
250       @post __post._totalSupply == _totalSupply - value
251       @post __post._balances[account] == _balances[account] - value
252      */
253     function _burnFrom(address account, uint256 value) internal {
254         _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(value);
255         _burn(account, value);
256         emit Approval(account, msg.sender, _allowed[account][msg.sender]);
257     }
258 }
```

File openzeppelin-solidity/contracts/token/ERC20/ERC20Mintable.sol

```
1  pragma solidity ^0.5.0;
2
3  import "./ERC20.sol";
4  import "../../access/roles/MinterRole.sol";
5
6  /**
7   * @title ERC20Mintable
8   * @dev ERC20 minting logic
9   */
10 contract ERC20Mintable is ERC20, MinterRole {
11     /**
12      * @dev Function to mint tokens
13      * @param to The address that will receive the minted tokens.
14      * @param value The amount of tokens to mint.
15      * @return A boolean that indicates if the operation was successful.
16      */
17     /*@CTK mint
18       @tag assume_completion
19       @post to != 0
20       @post __post._totalSupply == _totalSupply + value
21       @post __post._balances[to] == _balances[to] + value
22      */
23     function mint(address to, uint256 value) public onlyMinter returns (bool) {
24         _mint(to, value);
25         return true;
26     }
27 }
```

File openzeppelin-solidity/contracts/token/ERC20/ERC20Detailed.sol

```
1  pragma solidity ^0.5.0;
2
3  import "./IERC20.sol";
4
5  /**
6   * @title ERC20Detailed token
7   * @dev The decimals are only for visualization purposes.
8   * All the operations are done using the smallest and indivisible token unit,
9   * just as on Ethereum all the operations are done in wei.
10  */
11 contract ERC20Detailed is IERC20 {
12     string private _name;
```

```
13      string private _symbol;
14      uint8 private _decimals;
15
16      /*@CTK ERC20Detailed
17        @post __post._name == name
18        @post __post._symbol == symbol
19        @post __post._decimals == decimals
20       */
21      constructor (string memory name, string memory symbol, uint8 decimals) public {
22          _name = name;
23          _symbol = symbol;
24          _decimals = decimals;
25      }
26
27      /**
28       * @return the name of the token.
29       */
30      /*@CTK name
31        @post __return == _name
32       */
33      function name() public view returns (string memory) {
34          return _name;
35      }
36
37      /**
38       * @return the symbol of the token.
39       */
40      /*@CTK symbol
41        @post __return == _symbol
42       */
43      function symbol() public view returns (string memory) {
44          return _symbol;
45      }
46
47      /**
48       * @return the number of decimals of the token.
49       */
50      /*@CTK decimals
51        @post __return == _decimals
52       */
53      function decimals() public view returns (uint8) {
54          return _decimals;
55      }
56  }
```

File openzeppelin-solidity/contracts/token/ERC20/ERC20Capped.sol

```
1   pragma solidity ^0.5.0;
2
3   import "./ERC20Mintable.sol";
4
5   /**
6    * @title Capped token
7    * @dev Mintable token with a token cap.
8    */
9   contract ERC20Capped is ERC20Mintable {
10      uint256 private _cap;
11
12      /*@CTK ERC20Capped
```

```
13      @tag assume_completion
14      @post cap > 0
15      @post __post._cap == cap
16     */
17    constructor (uint256 cap) public {
18        require(cap > 0);
19        _cap = cap;
20    }
21
22    /**
23     * @return the cap for the token minting.
24     */
25    /*@CTK cap
26      @post __return == _cap
27     */
28    function cap() public view returns (uint256) {
29        return _cap;
30    }
31
32    /*@CTK _mint
33      @tag assume_completion
34      @post _totalSupply + value <= _cap
35      @post account != address(0)
36      @post __post._totalSupply == _totalSupply + value
37      @post __post._balances[account] == _balances[account] + value
38     */
39    function _mint(address account, uint256 value) internal {
40        require(totalSupply().add(value) <= _cap);
41        super._mint(account, value);
42    }
43 }
```

File openzeppelin-solidity/contracts/access/Roles.sol

```
1  pragma solidity ^0.5.0;
2
3  /**
4   * @title Roles
5   * @dev Library for managing addresses assigned to a Role.
6   */
7  library Roles {
8      struct Role {
9          mapping (address => bool) bearer;
10     }
11
12     /**
13      * @dev give an account access to this role
14      */
15     /*CTK add
16       @tag assume_completion
17       @post account != address(0)
18       @post !role.bearer[account]
19       @post __post.role.bearer[account]
20      */
21     function add(Role storage role, address account) internal {
22         require(account != address(0));
23         require(!has(role, account));
24
25         role.bearer[account] = true;
```

```
26      }
27
28      /**
29       * @dev remove an account's access to this role
30       */
31      /*CTK remove
32        @tag assume_completion
33        @post account != address(0)
34        @post role.bearer[account]
35        @post !__post.role.bearer[account]
36       */
37      function remove(Role storage role, address account) internal {
38          require(account != address(0));
39          require(has(role, account));
40
41          role.bearer[account] = false;
42      }
43
44      /**
45       * @dev check if an account has this role
46       * @return bool
47       */
48      /*@CTK has
49        @tag assume_completion
50        @post account != address(0)
51        @post __return == role.bearer[account]
52       */
53      function has(Role storage role, address account) internal view returns (bool) {
54          require(account != address(0));
55          return role.bearer[account];
56      }
57  }
```

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

```
1   pragma solidity ^0.5.0;
2
3   import "../access/roles/PauserRole.sol";
4
5   /**
6    * @title Pausable
7    * @dev Base contract which allows children to implement an emergency stop mechanism.
8    */
9   contract Pausable is PauserRole {
10      event Paused(address account);
11      event Unpaused(address account);
12
13      bool private _paused;
14
15      constructor () internal {
16          _paused = false;
17      }
18
19      /**
20       * @return true if the contract is paused, false otherwise.
21       */
22      /*@CTK paused
23        @post __return == _paused
24       */
```

```
25      function paused() public view returns (bool) {
26          return _paused;
27      }
28
29      /**
30       * @dev Modifier to make a function callable only when the contract is not paused.
31       */
32      modifier whenNotPaused() {
33          require(!_paused);
34          _;
35      }
36
37      /**
38       * @dev Modifier to make a function callable only when the contract is paused.
39       */
40      modifier whenPaused() {
41          require(_paused);
42          _;
43      }
44
45      /**
46       * @dev called by the owner to pause, triggers stopped state
47       */
48      /*@CTK pause
49        @tag assume_completion
50        @post !_paused
51        @post __post._paused
52       */
53      function pause() public onlyPauser whenNotPaused {
54          _paused = true;
55          emit Paused(msg.sender);
56      }
57
58      /**
59       * @dev called by the owner to unpause, returns to normal state
60       */
61      /*@CTK pause
62        @tag assume_completion
63        @post _paused
64        @post !__post._paused
65       */
66      function unpause() public onlyPauser whenPaused {
67          _paused = false;
68          emit Unpaused(msg.sender);
69      }
70  }
```

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

```
1   pragma solidity ^0.5.0;
2
3   /**
4    * @title Ownable
5    * @dev The Ownable contract has an owner address, and provides basic authorization
         control
6    * functions, this simplifies the implementation of "user permissions".
7    */
8   contract Ownable {
9       address private _owner;
```

```
10
11      event OwnershipTransferred(address indexed previousOwner, address indexed newOwner
            );
12
13      /**
14       * @dev The Ownable constructor sets the original `owner` of the contract to the
             sender
15       * account.
16       */
17      /*@CTK Ownable
18        @post __post._owner == msg.sender
19       */
20      constructor () internal {
21          _owner = msg.sender;
22          emit OwnershipTransferred(address(0), _owner);
23      }
24
25      /**
26       * @return the address of the owner.
27       */
28    /*@CTK owner
29      @post __return == _owner
30     */
31      function owner() public view returns (address) {
32          return _owner;
33      }
34
35      /**
36       * @dev Throws if called by any account other than the owner.
37       */
38      modifier onlyOwner() {
39          require(isOwner());
40          _;
41      }
42
43      /**
44       * @return true if `msg.sender` is the owner of the contract.
45       */
46    /*@CTK isOwner
47      @post __return == (msg.sender == _owner)
48     */
49      function isOwner() public view returns (bool) {
50          return msg.sender == _owner;
51      }
52
53      /**
54       * @dev Allows the current owner to relinquish control of the contract.
55       * @notice Renouncing to ownership will leave the contract without an owner.
56       * It will not be possible to call the functions with the `onlyOwner`
57       * modifier anymore.
58       */
59    /*@CTK renounceOwnership
60      @tag assume_completion
61      @post _owner == msg.sender
62      @post __post._owner == address(0)
63     */
64      function renounceOwnership() public onlyOwner {
65          emit OwnershipTransferred(_owner, address(0));
```

```
66          _owner = address(0);
67      }
68
69      /**
70       * @dev Allows the current owner to transfer control of the contract to a newOwner
             .
71       * @param newOwner The address to transfer ownership to.
72       */
73    /*@CTK transferOwnership
74      @tag assume_completion
75      @post _owner == msg.sender
76     */
77      function transferOwnership(address newOwner) public onlyOwner {
78          _transferOwnership(newOwner);
79      }
80
81      /**
82       * @dev Transfers control of the contract to a newOwner.
83       * @param newOwner The address to transfer ownership to.
84       */
85    /*@CTK _transferOwnership
86      @tag assume_completion
87      @post newOwner != address(0)
88      @post __post._owner == newOwner
89     */
90      function _transferOwnership(address newOwner) internal {
91          require(newOwner != address(0));
92          emit OwnershipTransferred(_owner, newOwner);
93          _owner = newOwner;
94      }
95  }
```

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms

CERTIK *label location*	Line 30-34 in File howtoread.sol

CERTIK *label*

```
30    /*@CTK FAIL "transferFrom to same address"
31        @tag assume_completion
32        @pre from == to
33        @post __post.allowed[from][msg.sender] ==
34    */
```

Raw code location	Line 35-41 in File howtoread.sol

Raw code

```
35    function transferFrom(address from, address to
          ) {
36        balances[from] = balances[from].sub(tokens
37        allowed[from][msg.sender] = allowed[from][
38        balances[to] = balances[to].add(tokens);
39        emit Transfer(from, to, tokens);
40        return true;
41    }
```

Counterexample	This code violates the specification

Initial environment

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          from = 0x0
5          to = 0x0
6          tokens = 0x6c
7      }
8      This = 0
                    /
53             balance: 0x0
54         }
55     }
56
```

Post environment

```
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

Static Analysis Request

Formal Verification Request 1

_transfer

08, Apr 2019
1869.53 ms

Line 50-56 in File OceanToken.sol

```
50    /*@CTK _transfer
51      @tag assume_completion
52      @pre msg.sender != _to
53      @post _to != address(0)
54      @post __post._balances[msg.sender] == _balances[msg.sender] - _value
55      @post __post._balances[_to] == _balances[_to] + _value
56     */
```

Line 57-69 in File OceanToken.sol

```
57    function transfer(
58        address _to,
59        uint256 _value
60    )
61        public
62        returns (bool)
63    {
64        bool success = super.transfer(_to, _value);
65        if (success) {
66            updateTokenHolders(msg.sender, _to);
67        }
68        return success;
69    }
```

The code meets the specification

Formal Verification Request 2

transfer_from

08, Apr 2019
1565.18 ms

Line 78-87 in File OceanToken.sol

```
78    /*@CTK transfer_from
79      @tag assume_completion
80      @pre _from != _to
81      @post _to != address(0)
82      @post _value <= _allowed[_from][msg.sender]
83      @post __post._balances[_from] == _balances[_from] - _value
84      @post __post._balances[_to] == _balances[_to] + _value
85      @post __post._allowed[_from][msg.sender] ==
86      _allowed[_from][msg.sender] - _value
87     */
```

Line 88-101 in File OceanToken.sol

```
88      function transferFrom(
89          address _from,
90          address _to,
91          uint256 _value
92      )
93          public
94          returns (bool)
95      {
96          bool success = super.transferFrom(_from, _to, _value);
97          if (success) {
98              updateTokenHolders(_from, _to);
99          }
100         return success;
101     }
```

The code meets the specification

Formal Verification Request 3

getAccountsLength

08, Apr 2019
39.7 ms

Line 145-149 in File OceanToken.sol

```
145     /*@CTK getAccountsLength
146       @tag assume_completion
147       @post _owner == msg.sender
148       @post __return == accounts.length
149      */
```

Line 150-157 in File OceanToken.sol

```
150     function getAccountsLength()
151         external
152         view
153         onlyOwner
154         returns (uint256)
155     {
156         return accounts.length;
157     }
```

The code meets the specification

Formal Verification Request 4

tryToAddTokenHolder

08, Apr 2019
6.57 ms

Line 183-188 in File OceanToken.sol

```
183     /*@CTK tryToAddTokenHolder
184       @tag assume_completion
185       @pre !tokenHolders[account] && _balances[account] > 0
186       @post __post.accounts[accounts.length] == account
187       @post __post.tokenHolders[account]
188      */
```

Line 189-199 in File OceanToken.sol

```
189     function tryToAddTokenHolder(
190         address account
191     )
192         private
193     {
194         if (!tokenHolders[account] && super.balanceOf(account) > 0)
195         {
196             accounts.push(account);
197             tokenHolders[account] = true;
198         }
199     }
```

The code meets the specification

Formal Verification Request 5

updateTokenHolders

08, Apr 2019
204.58 ms

Line 206-215 in File OceanToken.sol

```
206     /*@CTK updateTokenHolders
207       @tag assume_completion
208       @pre sender != receiver
209       @pre !tokenHolders[sender] && _balances[sender] > 0
210       @pre !tokenHolders[receiver] && _balances[receiver] > 0
211       @post __post.accounts[accounts.length] == sender
212       @post __post.tokenHolders[sender]
213       @post __post.accounts[accounts.length + 1] == receiver
214       @post __post.tokenHolders[receiver]
215      */
```

Line 216-224 in File OceanToken.sol

```
216     function updateTokenHolders(
217         address sender,
218         address receiver
219     )
220         private
221     {
222         tryToAddTokenHolder(sender);
223         tryToAddTokenHolder(receiver);
224     }
```

The code meets the specification

Formal Verification Request 6

Migrations

📅 08, Apr 2019
⏱ 6.86 ms

Line 7-9 in File Migrations.sol

```
7   /*@CTK Migrations
8     @post __post.owner == msg.sender
9    */
```

Line 10-12 in File Migrations.sol

```
10  constructor() public {
11      owner = msg.sender;
12  }
```

✅ The code meets the specification

Formal Verification Request 7

setCompleted

📅 08, Apr 2019
⏱ 11.14 ms

Line 18-21 in File Migrations.sol

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

Line 22-24 in File Migrations.sol

```
22  function setCompleted(uint completed) public restricted {
23      last_completed_migration = completed;
24  }
```

✅ The code meets the specification

Formal Verification Request 8

totalSupply

📅 08, Apr 2019
⏱ 6.86 ms

Line 30-32 in File ERC20.sol

```
30  /*@CTK totalSupply
31    @post __return == _totalSupply
32   */
```

Line 33-35 in File ERC20.sol

```
33    function totalSupply() public view returns (uint256) {
34        return _totalSupply;
35    }
```

The code meets the specification

Formal Verification Request 9

balanceOf

08, Apr 2019
7.7 ms

Line 42-44 in File ERC20.sol

```
42    /*@CTK balanceOf
43      @post __return == _balances[owner]
44     */
```

Line 45-47 in File ERC20.sol

```
45    function balanceOf(address owner) public view returns (uint256) {
46        return _balances[owner];
47    }
```

The code meets the specification

Formal Verification Request 10

allowance

08, Apr 2019
6.95 ms

Line 55-57 in File ERC20.sol

```
55    /*@CTK allowance
56      @post __return == _allowed[owner][spender]
57     */
```

Line 58-60 in File ERC20.sol

```
58    function allowance(address owner, address spender) public view returns (uint256) {
59        return _allowed[owner][spender];
60    }
```

The code meets the specification

Formal Verification Request 11

transfer

📅 08, Apr 2019
⏱ 267.24 ms

Line 67-74 in File ERC20.sol

```
67    /*@CTK transfer
68      @tag assume_completion
69      @pre msg.sender != to
70      @post to != address(0)
71      @post value <= _balances[msg.sender]
72      @post __post._balances[to] == _balances[to] + value
73      @post __post._balances[msg.sender] == _balances[msg.sender] - value
74     */
```

Line 75-78 in File ERC20.sol

```
75    function transfer(address to, uint256 value) public returns (bool) {
76        _transfer(msg.sender, to, value);
77        return true;
78    }
```

✅ The code meets the specification

Formal Verification Request 12

approve

📅 08, Apr 2019
⏱ 21.06 ms

Line 89-93 in File ERC20.sol

```
89    /*@CTK approve
90      @tag assume_completion
91      @post spender != address(0)
92      @post __post._allowed[msg.sender][spender] == value
93     */
```

Line 94-100 in File ERC20.sol

```
94    function approve(address spender, uint256 value) public returns (bool) {
95        require(spender != address(0));
96
97        _allowed[msg.sender][spender] = value;
98        emit Approval(msg.sender, spender, value);
99        return true;
100   }
```

✅ The code meets the specification

Formal Verification Request 13

transfer_from

08, Apr 2019
254.16 ms

Line 110-119 in File ERC20.sol

```
110     /*@CTK transfer_from
111       @tag assume_completion
112       @pre from != to
113       @post to != address(0)
114       @post value <= _allowed[from][msg.sender]
115       @post __post._balances[from] == _balances[from] - value
116       @post __post._balances[to] == _balances[to] + value
117       @post __post._allowed[from][msg.sender] ==
118       _allowed[from][msg.sender] - value
119      */
```

Line 120-125 in File ERC20.sol

```
120     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
121         _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
122         _transfer(from, to, value);
123         emit Approval(from, msg.sender, _allowed[from][msg.sender]);
124         return true;
125     }
```

The code meets the specification

Formal Verification Request 14

increaseAllowance

08, Apr 2019
58.46 ms

Line 137-142 in File ERC20.sol

```
137     /*@CTK increaseAllowance
138       @tag assume_completion
139       @post spender != address(0)
140       @post __post._allowed[msg.sender][spender] ==
141           _allowed[msg.sender][spender] + addedValue
142      */
```

Line 143-149 in File ERC20.sol

```
143     function increaseAllowance(address spender, uint256 addedValue) public returns (
            bool) {
144         require(spender != address(0));
145 
146         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].add(addedValue);
147         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
148         return true;
149     }
```

The code meets the specification

Formal Verification Request 15

decreaseAllowance

08, Apr 2019
62.49 ms

Line 161-166 in File ERC20.sol

```
161     /*@CTK decreaseAllowance
162       @tag assume_completion
163       @post spender != address(0)
164       @post __post._allowed[msg.sender][spender] ==
165           _allowed[msg.sender][spender] - subtractedValue
166      */
```

Line 167-173 in File ERC20.sol

```
167     function decreaseAllowance(address spender, uint256 subtractedValue) public
            returns (bool) {
168         require(spender != address(0));
169 
170         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].sub(
                subtractedValue);
171         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
172         return true;
173     }
```

The code meets the specification

Formal Verification Request 16

_transfer

08, Apr 2019
52.73 ms

Line 181-187 in File ERC20.sol

```
181     /*@CTK _transfer
182       @tag assume_completion
183       @pre from != to
184       @post to != address(0)
185       @post __post._balances[from] == _balances[from] - value
186       @post __post._balances[to] == _balances[to] + value
187      */
```

Line 188-194 in File ERC20.sol

```
188     function _transfer(address from, address to, uint256 value) internal {
189         require(to != address(0));
190 
191         _balances[from] = _balances[from].sub(value);
```

```
192         _balances[to] = _balances[to].add(value);
193         emit Transfer(from, to, value);
194     }
```

The code meets the specification

Formal Verification Request 17

_mint

08, Apr 2019
111.96 ms

Line 203-208 in File ERC20.sol

```
203     /*@CTK _mint
204       @tag assume_completion
205       @post account != 0
206       @post __post._totalSupply == _totalSupply + value
207       @post __post._balances[account] == _balances[account] + value
208      */
```

Line 209-215 in File ERC20.sol

```
209     function _mint(address account, uint256 value) internal {
210         require(account != address(0));
211 
212         _totalSupply = _totalSupply.add(value);
213         _balances[account] = _balances[account].add(value);
214         emit Transfer(address(0), account, value);
215     }
```

The code meets the specification

Formal Verification Request 18

_burn

08, Apr 2019
133.08 ms

Line 223-229 in File ERC20.sol

```
223     /*@CTK _burn
224       @tag assume_completion
225       @post account != 0
226       @post value <= _balances[account]
227       @post __post._totalSupply == _totalSupply - value
228       @post __post._balances[account] == _balances[account] - value
229      */
```

Line 230-236 in File ERC20.sol

```
230     function _burn(address account, uint256 value) internal {
231         require(account != address(0));
232 
233         _totalSupply = _totalSupply.sub(value);
234         _balances[account] = _balances[account].sub(value);
235         emit Transfer(account, address(0), value);
236     }
```

The code meets the specification

Formal Verification Request 19

_burnFrom

08, Apr 2019
315.61 ms

Line 246-252 in File ERC20.sol

```
246     /*@CTK _burnFrom
247       @tag assume_completion
248       @post value <= _allowed[account][msg.sender]
249       @post __post._allowed[account][msg.sender] == _allowed[account][msg.sender] -
              value
250       @post __post._totalSupply == _totalSupply - value
251       @post __post._balances[account] == _balances[account] - value
252      */
```

Line 253-257 in File ERC20.sol

```
253     function _burnFrom(address account, uint256 value) internal {
254         _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(value);
255         _burn(account, value);
256         emit Approval(account, msg.sender, _allowed[account][msg.sender]);
257     }
```

The code meets the specification

Formal Verification Request 20

mint

08, Apr 2019
428.84 ms

Line 17-22 in File ERC20Mintable.sol

```
17     /*@CTK mint
18       @tag assume_completion
19       @post to != 0
20       @post __post._totalSupply == _totalSupply + value
21       @post __post._balances[to] == _balances[to] + value
22      */
```

Line 23-26 in File ERC20Mintable.sol

```
23    function mint(address to, uint256 value) public onlyMinter returns (bool) {
24        _mint(to, value);
25        return true;
26    }
```

The code meets the specification

Formal Verification Request 21

ERC20Detailed

08, Apr 2019
14.7 ms

Line 16-20 in File ERC20Detailed.sol

```
16    /*@CTK ERC20Detailed
17      @post __post._name == name
18      @post __post._symbol == symbol
19      @post __post._decimals == decimals
20     */
```

Line 21-25 in File ERC20Detailed.sol

```
21    constructor (string memory name, string memory symbol, uint8 decimals) public {
22        _name = name;
23        _symbol = symbol;
24        _decimals = decimals;
25    }
```

The code meets the specification

Formal Verification Request 22

name

08, Apr 2019
7.47 ms

Line 30-32 in File ERC20Detailed.sol

```
30    /*@CTK name
31      @post __return == _name
32     */
```

Line 33-35 in File ERC20Detailed.sol

```
33    function name() public view returns (string memory) {
34        return _name;
35    }
```

The code meets the specification

Formal Verification Request 23

symbol

📅 08, Apr 2019
⏱ 8.53 ms

Line 40-42 in File ERC20Detailed.sol

```
40  /*@CTK symbol
41    @post __return == _symbol
42   */
```

Line 43-45 in File ERC20Detailed.sol

```
43  function symbol() public view returns (string memory) {
44      return _symbol;
45  }
```

✅ The code meets the specification

Formal Verification Request 24

decimals

📅 08, Apr 2019
⏱ 6.71 ms

Line 50-52 in File ERC20Detailed.sol

```
50  /*@CTK decimals
51    @post __return == _decimals
52   */
```

Line 53-55 in File ERC20Detailed.sol

```
53  function decimals() public view returns (uint8) {
54      return _decimals;
55  }
```

✅ The code meets the specification

Formal Verification Request 25

ERC20Capped

📅 08, Apr 2019
⏱ 16.87 ms

Line 12-16 in File ERC20Capped.sol

```
12  /*@CTK ERC20Capped
13    @tag assume_completion
14    @post cap > 0
15    @post __post._cap == cap
16   */
```

Line 17-20 in File ERC20Capped.sol

```
17    constructor (uint256 cap) public {
18        require(cap > 0);
19        _cap = cap;
20    }
```

The code meets the specification

Formal Verification Request 26

cap

08, Apr 2019
5.98 ms

Line 25-27 in File ERC20Capped.sol

```
25    /*@CTK cap
26      @post __return == _cap
27     */
```

Line 28-30 in File ERC20Capped.sol

```
28    function cap() public view returns (uint256) {
29        return _cap;
30    }
```

The code meets the specification

Formal Verification Request 27

_mint

08, Apr 2019
515.64 ms

Line 32-38 in File ERC20Capped.sol

```
32    /*@CTK _mint
33      @tag assume_completion
34      @post _totalSupply + value <= _cap
35      @post account != address(0)
36      @post __post._totalSupply == _totalSupply + value
37      @post __post._balances[account] == _balances[account] + value
38     */
```

Line 39-42 in File ERC20Capped.sol

```
39    function _mint(address account, uint256 value) internal {
40        require(totalSupply().add(value) <= _cap);
41        super._mint(account, value);
42    }
```

The code meets the specification

Formal Verification Request 28

has

08, Apr 2019
16.77 ms

Line 48-52 in File Roles.sol

```
48    /*@CTK has
49      @tag assume_completion
50      @post account != address(0)
51      @post __return == role.bearer[account]
52     */
```

Line 53-56 in File Roles.sol

```
53    function has(Role storage role, address account) internal view returns (bool) {
54        require(account != address(0));
55        return role.bearer[account];
56    }
```

The code meets the specification

Formal Verification Request 29

paused

08, Apr 2019
8.59 ms

Line 22-24 in File Pausable.sol

```
22    /*@CTK paused
23      @post __return == _paused
24     */
```

Line 25-27 in File Pausable.sol

```
25    function paused() public view returns (bool) {
26        return _paused;
27    }
```

The code meets the specification

Formal Verification Request 30

pause

08, Apr 2019
139.27 ms

Line 48-52 in File Pausable.sol

```
48    /*@CTK pause
49      @tag assume_completion
50      @post !_paused
51      @post __post._paused
52     */
```

Line 53-56 in File Pausable.sol

```
53    function pause() public onlyPauser whenNotPaused {
54        _paused = true;
55        emit Paused(msg.sender);
56    }
```

The code meets the specification

Formal Verification Request 31

pause

08, Apr 2019
75.94 ms

Line 61-65 in File Pausable.sol

```
61    /*@CTK pause
62      @tag assume_completion
63      @post _paused
64      @post !__post._paused
65     */
```

Line 66-69 in File Pausable.sol

```
66    function unpause() public onlyPauser whenPaused {
67        _paused = false;
68        emit Unpaused(msg.sender);
69    }
```

The code meets the specification

Formal Verification Request 32

Ownable

08, Apr 2019
8.4 ms

Line 17-19 in File Ownable.sol

```
17    /*@CTK Ownable
18      @post __post._owner == msg.sender
19     */
```

Line 20-23 in File Ownable.sol

```
20      constructor () internal {
21          _owner = msg.sender;
22          emit OwnershipTransferred(address(0), _owner);
23      }
```

The code meets the specification

Formal Verification Request 33

owner

08, Apr 2019
6.89 ms

Line 28-30 in File Ownable.sol

```
28    /*@CTK owner
29      @post __return == _owner
30     */
```

Line 31-33 in File Ownable.sol

```
31      function owner() public view returns (address) {
32          return _owner;
33      }
```

The code meets the specification

Formal Verification Request 34

isOwner

08, Apr 2019
10.88 ms

Line 46-48 in File Ownable.sol

```
46    /*@CTK isOwner
47      @post __return == (msg.sender == _owner)
48     */
```

Line 49-51 in File Ownable.sol

```
49      function isOwner() public view returns (bool) {
50          return msg.sender == _owner;
51      }
```

The code meets the specification

Formal Verification Request 35

renounceOwnership

📅 08, Apr 2019
⏱ 30.09 ms

Line 59-63 in File Ownable.sol

```
59  /*@CTK renounceOwnership
60    @tag assume_completion
61    @post _owner == msg.sender
62    @post __post._owner == address(0)
63   */
```

Line 64-67 in File Ownable.sol

```
64    function renounceOwnership() public onlyOwner {
65        emit OwnershipTransferred(_owner, address(0));
66        _owner = address(0);
67    }
```

✅ The code meets the specification

Formal Verification Request 36

transferOwnership

📅 08, Apr 2019
⏱ 71.75 ms

Line 73-76 in File Ownable.sol

```
73  /*@CTK transferOwnership
74    @tag assume_completion
75    @post _owner == msg.sender
76   */
```

Line 77-79 in File Ownable.sol

```
77    function transferOwnership(address newOwner) public onlyOwner {
78        _transferOwnership(newOwner);
79    }
```

✅ The code meets the specification

Formal Verification Request 37

_transferOwnership

📅 08, Apr 2019
⏱ 1.53 ms

Line 85-89 in File Ownable.sol

```
85  /*@CTK _transferOwnership
86    @tag assume_completion
87    @post newOwner != address(0)
88    @post __post._owner == newOwner
89   */
```

Line 90-94 in File Ownable.sol

```
90    function _transferOwnership(address newOwner) internal {
91        require(newOwner != address(0));
92        emit OwnershipTransferred(_owner, newOwner);
93        _owner = newOwner;
94    }
```

The code meets the specification