

CERTIK AUDIT REPORT FOR XUSB



Request Date: 2019-08-05
Revision Date: 2019-08-06
Platform Name: Ethereum



Contents

Disclaimer	1
About CertiK	2
Exective Summary	3
Vulnerability Classification	3
Testing Summary	4
Audit Score	4
Type of Issues	4
Vulnerability Details	5
Manual Review Notes	6
Static Analysis Results	8
Formal Verification Results	10
How to read	10
Source Code with CertiK Labels	26

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 XUSB(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.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 6.2B in assets.

For more information: <https://certik.org/>

Exective Summary

This report has been prepared as the product of the Smart Contract Audit request by XUSB. This audit was conducted to discover issues and vulnerabilities in the source code of XUSB's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review, a comprehensive examination has been performed. The auditing process pays special attention to the following considerations.

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

Vulnerability Classification

For every issue 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 conditions, 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 vulnerabilities, but no concern found yet.

Testing Summary

PASS

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

Aug 05, 2019



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	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.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **Migrations.sol**
1c4e30fd3aa765cb0ee259a29dead71c1c99888dcc7157c25df3405802cf5b09
- **StableToken.sol**
d6601e40330b07ffc1f5e8b6d53272d34cdd67f57d1a00e4242f60b409bd4a46
- **StableTokenTimelock.sol**
7c2d26ad383170bdc9570dea804727664d59975b7238d79b383b6feb5a614abb

Summary

CertiK team is invited by the XUSB team to audit the design and implementations of its to be released ERC20 based smart contract, and the source code has been analyzed under different perspectives and with different tools such as CertiK formal verification checking as well as manual reviews by smart contract experts. We have been actively interacting with client-side engineers when there was any potential loopholes or recommended design changes during the audit process, and The XUSB team has been actively giving us updates for the source code and feedback about the business logics.

At this point the XUSB team didn't provide other repositories sources as testing and documentation reference. We recommend having more unit tests coverage together with documentation to simulate potential use cases and walk through the functionalities to token holders, especially those super admin privileges that may impact the decentralized nature. Meanwhile, we are glad to see that XUSB team takes transparency seriously (i.e. supply and issue mechanism for each party is strictly written and followed to against any potential mischievous behaviors) and implement the lockup schedule with great care. However on the other hand, we recommend to decouple such features into different components so as to have a much cleaner token contract without any add-ons that purely for the purpose of lockup.

Overall we found the contracts follow good practices, with reasonable amount of features on top of the ERC20 related to administrative controls by the token issuer. With the final update of source code and delivery of the audit report, we conclude that the contract is not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend seeking multiple opinions, more test coverage and sandbox deployments before the mainnet release.

Recommendations

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes.

StableTokenTimelock.sol

- **MINOR** `transfer()`: SafeMath library is recommended to avoid accidental overflow problems.
- **INFO** `transfer()`: Recommend moving event emission after `_token.safeTransfer(recipient, amount)`.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File StableToken.sol

```
1 pragma solidity ^0.5.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

INSECURE_COMPILER_VERSION

Line 1 in File StableTokenTimelock.sol


```
1 pragma solidity ^0.5.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

TIMESTAMP_DEPENDENCY

Line 111 in File StableTokenTimelock.sol


```
111     require(block.timestamp >= _releaseTime, "TokenTimelock: current time is before  
        release time");
```

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

INSECURE_COMPILER_VERSION

Line 1 in File Migrations.sol

```
1 pragma solidity >=0.4.21 <0.6.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

INSECURE_COMPILER_VERSION

Line 1 in File ERC20.sol

```
1 pragma solidity ^0.5.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

INSECURE_COMPILER_VERSION

Line 1 in File ERC20Mintable.sol

```
1 pragma solidity ^0.5.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

INSECURE_COMPILER_VERSION

Line 1 in File ERC20Detailed.sol

```
1 pragma solidity ^0.5.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

INSECURE_COMPILER_VERSION

Line 1 in File ERC20Capped.sol

```
1 pragma solidity ^0.5.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

INSECURE_COMPILER_VERSION

Line 1 in File Roles.sol

```
1 pragma solidity ^0.5.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

INSECURE_COMPILER_VERSION

Line 1 in File Ownable.sol

```
1 pragma solidity ^0.5.0;
```

 Only these compiler versions are safe to compile your code: 0.5.9

Formal Verification Results

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

Formal Verification Request 1

StableToken

📅 05, Aug 2019

🕒 37.1 ms

Line 8-12 in File StableToken.sol

```
8      /*@CTK StableToken
9         @post __post._name == name
10        @post __post._symbol == symbol
11        @post __post._decimals == decimals
12      */
```

Line 13-22 in File StableToken.sol

```
13      constructor(
14          string memory name,
15          string memory symbol,
16          uint8 decimals
17      )
18          ERC20Detailed(name, symbol, decimals)
19      public
20      {
21          // _mint(msg.sender, initSupply);
22      }
```

✅ The code meets the specification.

Formal Verification Request 2

StableTokenTimelock

📅 05, Aug 2019

🕒 32.29 ms

Line 23-30 in File StableTokenTimelock.sol

```
23      /*@CTK StableTokenTimelock
24         @tag assume_completion
25         @post releaseTime > block.timestamp
26         @post __post._token == token
27         @post __post._beneficiary == beneficiary
28         @post __post._releaseTime == releaseTime
29         @post __post._lockupSupply == lockupSupply
30      */
```

Line 31-44 in File StableTokenTimelock.sol

```
31      constructor (
32          IERC20 token,
33          address beneficiary,
34          uint256 releaseTime,
35          uint256 lockupSupply
36      ) public {
37          // solhint-disable-next-line not-rely-on-time
38          require(releaseTime > block.timestamp, "StableTokenTimelock: release time is
              before current time");
```


```
39      // require(_token.balanceOf(address(this)) >= lockupSupply, "  
      StableTokenTimelock: Lockup supply exceeds token total supply");  
40      _token = token;  
41      _beneficiary = beneficiary;  
42      _releaseTime = releaseTime;  
43      _lockupSupply = lockupSupply;  
44  }
```

✓ The code meets the specification.

Formal Verification Request 3

beneficiary

 05, Aug 2019

 5.96 ms

Line 56-58 in File StableTokenTimelock.sol

```
56      /*@CTK beneficiary  
57      @post __return == _beneficiary  
58      */
```

Line 59-61 in File StableTokenTimelock.sol


```
59      function beneficiary() public view returns (address) {  
60          return _beneficiary;  
61      }
```

✓ The code meets the specification.

Formal Verification Request 4

releaseTime

 05, Aug 2019

 6.15 ms

Line 66-68 in File StableTokenTimelock.sol

```
66      /*@CTK releaseTime  
67      @post __return == _releaseTime  
68      */
```

Line 69-71 in File StableTokenTimelock.sol

```
69      function releaseTime() public view returns (uint256) {  
70          return _releaseTime;  
71      }
```

✓ The code meets the specification.

Formal Verification Request 5

lockupSupply

📅 05, Aug 2019

🕒 5.94 ms

Line 76-78 in File StableTokenTimelock.sol

```
76  /*@CTK lockupSupply
77      @post __return == _lockupSupply
78  */
```

Line 79-81 in File StableTokenTimelock.sol

```
79  function lockupSupply() public view returns (uint256) {
80      return _lockupSupply;
81  }
```

✅ The code meets the specification.

Formal Verification Request 6

Migrations

📅 05, Aug 2019

🕒 5.47 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

📅 05, Aug 2019

🕒 9.28 ms

Line 18-21 in File Migrations.sol

```
18  /*@CTK setCompleted
19      @pre owner == msg.sender
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

 05, Aug 2019

 5.33 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

 05, Aug 2019

 6.04 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

 05, Aug 2019

 5.69 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

📅 05, Aug 2019

🕒 197.83 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

📅 05, Aug 2019

🕒 17.0 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

📅 05, Aug 2019

🕒 195.06 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 (
121         bool) {
122         _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
123         _transfer(from, to, value);
124         emit Approval(from, msg.sender, _allowed[from][msg.sender]);
125         return true;
126     }

```

✓ The code meets the specification.

Formal Verification Request 14

increaseAllowance

📅 05, Aug 2019

🕒 46.68 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 (
144         bool) {
145         require(spender != address(0));
146
147         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].add(addedValue);
148         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
149         return true;
150     }
```

✓ The code meets the specification.

Formal Verification Request 15

decreaseAllowance

📅 05, Aug 2019

🕒 48.18 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
168         returns (bool) {
169         require(spender != address(0));
170
171         _allowed[msg.sender][spender] = _allowed[msg.sender][spender].sub(
172             subtractedValue);
173         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
174         return true;
175     }
```

✓ The code meets the specification.

Formal Verification Request 16

_transfer

📅 05, Aug 2019

🕒 41.07 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

 05, Aug 2019

 74.22 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

 05, Aug 2019

 106.99 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

📅 05, Aug 2019

⌚ 251.62 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

📅 05, Aug 2019

⌚ 313.75 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

 05, Aug 2019

 9.4 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

 05, Aug 2019

 5.81 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

05, Aug 2019

6.43 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

05, Aug 2019

5.06 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

05, Aug 2019

15.26 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

 05, Aug 2019

 5.38 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

 05, Aug 2019

 464.09 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

📅 05, Aug 2019

🕒 13.03 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

Ownable

📅 05, Aug 2019

🕒 6.63 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 30

owner

📅 05, Aug 2019

🕒 6.3 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 31

isOwner

 05, Aug 2019

 5.95 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 32

renounceOwnership

 05, Aug 2019

 24.52 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 33

transferOwnership

 05, Aug 2019

 54.25 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 34

_transferOwnership

 05, Aug 2019

 1.17 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.

Source Code with CertiK Labels

File StableToken.sol

```

1  pragma solidity ^0.5.0;
2
3  import "openzeppelin-solidity/contracts/token/ERC20/ERC20.sol";
4  import "openzeppelin-solidity/contracts/token/ERC20/ERC20Detailed.sol";
5  import "openzeppelin-solidity/contracts/token/ERC20/ERC20Mintable.sol";
6
7  contract StableToken is ERC20, ERC20Detailed, ERC20Mintable {
8      /*@CTK StableToken
9          @post __post.__name == name
10         @post __post.__symbol == symbol
11         @post __post.__decimals == decimals
12         */
13     constructor(
14         string memory name,
15         string memory symbol,
16         uint8 decimals
17     )
18         ERC20Detailed(name, symbol, decimals)
19     public
20     {
21         // _mint(msg.sender, initSupply);
22     }
23 }

```

File StableTokenTimelock.sol

```

1  pragma solidity ^0.5.0;
2
3  import "openzeppelin-solidity/contracts/ownership/Ownable.sol";
4  import "openzeppelin-solidity/contracts/token/ERC20/SafeERC20.sol";
5
6  contract StableTokenTimelock is Ownable {
7      using SafeERC20 for IERC20;
8
9      // ERC20 basic token contract being held
10     IERC20 private _token;
11
12     // beneficiary of tokens after they are released
13     address private _beneficiary;
14
15     // timestamp when token release is enabled
16     uint256 private _releaseTime;
17
18     // lockup supply
19     uint256 private _lockupSupply;
20
21     event TimelockTransfer(address receiver, uint256 availableAmount, uint256 amount);
22
23     /*@CTK StableTokenTimelock
24         @tag assume_completion
25         @post releaseTime > block.timestamp
26         @post __post.__token == token
27         @post __post.__beneficiary == beneficiary
28         @post __post.__releaseTime == releaseTime
29         @post __post.__lockupSupply == lockupSupply

```

```

30  */
31  constructor (
32      IERC20 token,
33      address beneficiary,
34      uint256 releaseTime,
35      uint256 lockupSupply
36  ) public {
37      // solhint-disable-next-line not-rely-on-time
38      require(releaseTime > block.timestamp, "StableTokenTimelock: release time is
        before current time");
39      // require(_token.balanceOf(address(this)) >= lockupSupply, "
        StableTokenTimelock: Lockup supply exceeds token total supply");
40      _token = token;
41      _beneficiary = beneficiary;
42      _releaseTime = releaseTime;
43      _lockupSupply = lockupSupply;
44  }
45
46  /**
47   * @return the token being held.
48   */
49  function token() public view returns (IERC20) {
50      return _token;
51  }
52
53  /**
54   * @return the beneficiary of the tokens.
55   */
56  /*@CTK beneficiary
57   @post __return == _beneficiary
58   */
59  function beneficiary() public view returns (address) {
60      return _beneficiary;
61  }
62
63  /**
64   * @return the time when the tokens are released.
65   */
66  /*@CTK releaseTime
67   @post __return == _releaseTime
68   */
69  function releaseTime() public view returns (uint256) {
70      return _releaseTime;
71  }
72
73  /**
74   * @return the lockup supply.
75   */
76  /*@CTK lockupSupply
77   @post __return == _lockupSupply
78   */
79  function lockupSupply() public view returns (uint256) {
80      return _lockupSupply;
81  }
82
83  /**
84   * @notice See 'IERC20.transfer'.
85   */

```

```

86     * Requirements:
87     *
88     * - 'recipient' cannot be the zero address.
89     * - the caller must have a balance of at least 'amount'.
90     */
91     /*CTK transfer
92     @tag assume_completion
93     @post _owner == msg.sender
94     */
95     function transfer(address recipient, uint256 amount) public onlyOwner returns (
96         bool) {
97         uint256 availableAmount = _token.balanceOf(address(this)) - _lockupSupply;
98         emit TimelockTransfer(recipient, availableAmount, amount);
99         require(availableAmount >= amount, "StableTokenTimelock: Insufficient funds");
100         _token.safeTransfer(recipient, amount);
101     }
102     /**
103     * @notice Transfers tokens held by timelock to beneficiary.
104     */
105     /*CTK release
106     @tag assume_completion
107     @post _owner == msg.sender
108     */
109     function release() public {
110         // solhint-disable-next-line not-rely-on-time
111         require(block.timestamp >= _releaseTime, "TokenTimelock: current time is before
112             release time");
113
114         uint256 amount = _token.balanceOf(address(this));
115         require(amount > 0, "TokenTimelock: no tokens to release");
116
117         _token.safeTransfer(_beneficiary, amount);
118     }

```

File Migrations.sol

```

1  pragma solidity >=0.4.21 <0.6.0;
2
3  contract Migrations {
4      address public owner;
5      uint public last_completed_migration;
6
7      /*@CTK Migrations
8      @post __post.owner == msg.sender
9      */
10     constructor() public {
11         owner = msg.sender;
12     }
13
14     modifier restricted() {
15         if (msg.sender == owner) _;
16     }
17
18     /*@CTK setCompleted
19     @pre owner == msg.sender
20     @post __post.last_completed_migration == completed
21     */

```

```

22 function setCompleted(uint completed) public restricted {
23     last_completed_migration = completed;
24 }
25
26 function upgrade(address new_address) public restricted {
27     Migrations upgraded = Migrations(new_address);
28     upgraded.setCompleted(last_completed_migration);
29 }
30 }

```

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

```

1 pragma solidity ^0.5.0;
2
3 import "./IERC20.sol";
4 import "../math/SafeMath.sol";
5
6 /**
7  * @title Standard ERC20 token
8  *
9  * @dev Implementation of the basic standard token.
10 * https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
11 * Originally based on code by FirstBlood:
12 * https://github.com/Firstbloodio/token/blob/master/smart\_contract/FirstBloodToken.sol
13 *
14 * This implementation emits additional Approval events, allowing applications to
15 * reconstruct the allowance status for
16 * all accounts just by listening to said events. Note that this isn't required by the
17 * specification, and other
18 * compliant implementations may not do it.
19 */
20 contract ERC20 is IERC20 {
21     using SafeMath for uint256;
22
23     mapping (address => uint256) private _balances;
24
25     mapping (address => mapping (address => uint256)) private _allowed;
26
27     uint256 private _totalSupply;
28
29     /**
30      * @dev Total number of tokens in existence
31      */
32     /*@CTK totalSupply
33      @post __return == _totalSupply
34      */
35     function totalSupply() public view returns (uint256) {
36         return _totalSupply;
37     }
38
39     /**
40      * @dev Gets the balance of the specified address.
41      * @param owner The address to query the balance of.
42      * @return An uint256 representing the amount owned by the passed address.
43      */
44     /*@CTK balanceOf
45      @post __return == _balances[owner]
46      */

```

```

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
54         spender.
55  */
56 /**@CTK allowance
57  @post __return == _allowed[owner][spender]
58  */
59 function allowance(address owner, address spender) public view returns (uint256) {
60     return _allowed[owner][spender];
61 }
62
63 /**
64  * @dev Transfer token for a specified address
65  * @param to The address to transfer to.
66  * @param value The amount to be transferred.
67  */
68 /**@CTK transfer
69  @tag assume_completion
70  @pre msg.sender != to
71  @post to != address(0)
72  @post value <= _balances[msg.sender]
73  @post __post._balances[to] == _balances[to] + value
74  @post __post._balances[msg.sender] == _balances[msg.sender] - value
75  */
76 function transfer(address to, uint256 value) public returns (bool) {
77     _transfer(msg.sender, to, value);
78     return true;
79 }
80
81 /**
82  * @dev Approve the passed address to spend the specified amount of tokens on
83         behalf of msg.sender.
84  * Beware that changing an allowance with this method brings the risk that someone
85         may use both the old
86  * and the new allowance by unfortunate transaction ordering. One possible
87         solution to mitigate this
88  * race condition is to first reduce the spender's allowance to 0 and set the
89         desired value afterwards:
90  * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
91  * @param spender The address which will spend the funds.
92  * @param value The amount of tokens to be spent.
93  */
94 /**@CTK approve
95  @tag assume_completion
96  @post spender != address(0)
97  @post __post._allowed[msg.sender][spender] == value
98  */
99 function approve(address spender, uint256 value) public returns (bool) {
100     require(spender != address(0));
101     _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
105  * per the specification,
106  * and other compliant implementations may not emit the event.
107  * @param from address The address which you want to send tokens from
108  * @param to address The address which you want to transfer to
109  * @param value uint256 the amount of tokens to be transferred
110  */
111 /*@CTK transfer_from
112  @tag assume_completion
113  @pre from != to
114  @post to != address(0)
115  @post value <= _allowed[from][msg.sender]
116  @post __post._balances[from] == _balances[from] - value
117  @post __post._balances[to] == _balances[to] + value
118  @post __post._allowed[from][msg.sender] ==
119  _allowed[from][msg.sender] - value
120 */
121 function transferFrom(address from, address to, uint256 value) public returns (
122     bool) {
123     _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
124     _transfer(from, to, value);
125     emit Approval(from, msg.sender, _allowed[from][msg.sender]);
126     return true;
127 }
128
129 /**
130  * @dev Increase the amount of tokens that an owner allowed to a spender.
131  * approve should be called when allowed[_spender] == 0. To increment
132  * allowed value is better to use this function to avoid 2 calls (and wait until
133  * the first transaction is mined)
134  * From MonolithDAO Token.sol
135  * Emits an Approval event.
136  * @param spender The address which will spend the funds.
137  * @param addedValue The amount of tokens to increase the allowance by.
138  */
139 /*@CTK increaseAllowance
140  @tag assume_completion
141  @post spender != address(0)
142  @post __post._allowed[msg.sender][spender] ==
143  _allowed[msg.sender][spender] + addedValue
144 */
145 function increaseAllowance(address spender, uint256 addedValue) public returns (
146     bool) {
147     require(spender != address(0));
148
149     _allowed[msg.sender][spender] = _allowed[msg.sender][spender].add(addedValue);
150     emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
151     return true;
152 }
153
154 /**
155  * @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
168     returns (bool) {
169     require(spender != address(0));
170
171     _allowed[msg.sender][spender] = _allowed[msg.sender][spender].sub(
172         subtractedValue);
173     emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
174     return true;
175 }
176
177 /**
178  * @dev Transfer token for a specified addresses
179  * @param from The address to transfer from.
180  * @param to The address to transfer to.
181  * @param value The amount to be transferred.
182  */
183 /*@CTK _transfer
184   @tag assume_completion
185   @pre from != to
186   @post to != address(0)
187   @post __post._balances[from] == _balances[from] - value
188   @post __post._balances[to] == _balances[to] + value
189 */
190 function _transfer(address from, address to, uint256 value) internal {
191     require(to != address(0));
192
193     _balances[from] = _balances[from].sub(value);
194     _balances[to] = _balances[to].add(value);
195     emit Transfer(from, to, value);
196 }
197
198 /**
199  * @dev Internal function that mints an amount of the token and assigns it to
200  * an account. This encapsulates the modification of balances such that the
201  * proper events are emitted.
202  * @param account The account that will receive the created tokens.
203  * @param value The amount that will be created.
204  */
205 /*@CTK _mint
206   @tag assume_completion
207   @post account != 0
208   @post __post._totalSupply == _totalSupply + value
209   @post __post._balances[account] == _balances[account] + value
210 */

```

```

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/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
6   * control
7   * functions, this simplifies the implementation of "user permissions".
8   */
9  contract Ownable {
10     address private _owner;
11
12     event OwnershipTransferred(address indexed previousOwner, address indexed newOwner
13         );
14
15     /**
16      * @dev The Ownable constructor sets the original 'owner' of the contract to the
17      * sender
18      * account.
19      */
20     /*@CTK Ownable
21         @post __post._owner == msg.sender
22     */
23     constructor () internal {
24         _owner = msg.sender;
25         emit OwnershipTransferred(address(0), _owner);
26     }
27
28     /**
29      * @return the address of the owner.
30      */
31     /*@CTK owner
32         @post __return == _owner
33     */
34     function owner() public view returns (address) {
35         return _owner;
36     }
37
38     /**
39      * @dev Throws if called by any account other than the owner.
40      */
41     modifier onlyOwner() {
42         require(isOwner());
43         _;
44     }
45 }

```

```

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

```