CertiK Audit Report For Taxa

Request Date: 2019-06-26
Revision Date: 2019-07-02
Platform Name: Ethereum

CERTIK

Contents

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 Taxa(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 1.4B in assets.

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

Exective Summary

This report has been prepared as product of the Smart Contract Audit request by Taxa. This audit was conducted to discover issues and vulnerabilities in the source code of Taxa'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 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

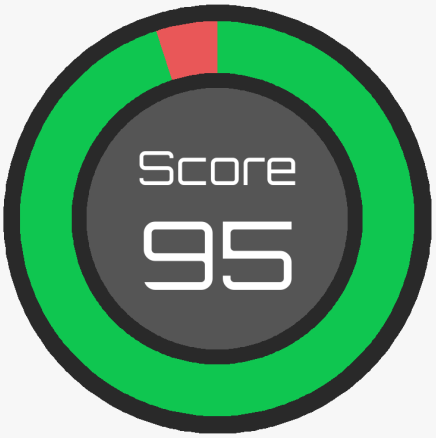
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

Testing Summary

PASS

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

Jul 02, 2019



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.	2	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.	13	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

- **TaxaLockFoundation.sol**
  `edd4a1fb9412571a10a1b62d6faba74d41ac1282402f18e6f65bed350c0bc48b`
- **TaxaToken.sol**
  `ec9511b7a51addf40a189f0147059d8658e09e4912002d7e887b312d87e02fb2`

Summary

CertiK was chosen by Taxa to audit the design and implementation of its soon to be released smart contract `TaxaNetworkToken`. To ensure comprehensive protection,the source code has been analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space.

Overall we found the smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and 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 to seek multiple opinions, keep improving the codebase, and more test coverage and sandbox deployments before the mainnet release.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File TaxaToken.sol

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

Version to compile has the following bug: 0.4.18: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.19: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.20: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.21: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.22: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, OneOfTwoConstructorsSkipped 0.4.23: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 1 in File TaxaLockFoundation.sol

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

Version to compile has the following bug: 0.4.23: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

TIMESTAMP_DEPENDENCY

Line 317 in File TaxaLockFoundation.sol

```
317     if (block.timestamp < cliff) {
```

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

TIMESTAMP_DEPENDENCY

Line 319 in File TaxaLockFoundation.sol

```
319     } else if (block.timestamp >= start.add(duration) || revoked) {
```

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

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
...
53             balance: 0x0
54         }
55     }
56
```

Post environment

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

Formal Verification Request 1

Method will not encounter an assertion failure.

02, Jul 2019
22.24 ms

Line 8 in File TaxaToken.sol

```
8     //@CTK FAIL NO_ASF
```

Line 16-23 in File TaxaToken.sol

```
16    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
17        if (a == 0) {
18            return 0;
19        }
20        uint256 c = a * b;
21        assert(c / a == b);
22        return c;
23    }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          a = 2
5          b = 156
6      }
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14           "gas": 0,
15           "sender": 0,
16           "value": 0
17         }
18     }
19     Other = {
20         __return = 0
21         block = {
22           "number": 0,
23           "timestamp": 0
24         }
25     }
26     Address_Map = [
27       {
28         "key": "ALL_OTHERS",
29         "value": "EmptyAddress"
30       }
31     ]
32
33 Function invocation is reverted.
```

Formal Verification Request 2

SafeMath mul

02, Jul 2019
294.28 ms

Line 9-15 in File TaxaToken.sol

```
9   /*@CTK "SafeMath mul"
10    @post ((a > 0) && (((a * b) / a) != b)) == (__reverted)
11    @post !__reverted -> __return == a * b
12    @post !__reverted == !__has_overflow
13    @post !__reverted -> !(__has_assertion_failure)
14    @post !(__has_buf_overflow)
15  */
```

Line 16-23 in File TaxaToken.sol

```
16  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
17      if (a == 0) {
18          return 0;
19      }
20      uint256 c = a * b;
21      assert(c / a == b);
22      return c;
23  }
```

The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.

02, Jul 2019
5.26 ms

Line 25 in File TaxaToken.sol

```
25  //@CTK FAIL NO_ASF
```

Line 33-38 in File TaxaToken.sol

```
33  function div(uint256 a, uint256 b) internal pure returns (uint256) {
34      // assert(b > 0); // Solidity automatically throws when dividing by 0
35      uint256 c = a / b;
36      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
37      return c;
38  }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          a = 0
5          b = 0
6      }
7      Internal = {
```

```
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = 0
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 4

SafeMath div

02, Jul 2019
0.86 ms

Line 26-32 in File TaxaToken.sol

```
    /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
    */
```

Line 33-38 in File TaxaToken.sol

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

The code meets the specification.

Formal Verification Request 5

Method will not encounter an assertion failure.

02, Jul 2019
11.28 ms

Line 40 in File TaxaToken.sol

```
40      //@CTK FAIL NO_ASF
```

Line 48-51 in File TaxaToken.sol

```
48      function sub(uint256 a, uint256 b) internal pure returns (uint256) {
49          assert(b <= a);
50          return a - b;
51      }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          a = 0
5          b = 1
6      }
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14           "gas": 0,
15           "sender": 0,
16           "value": 0
17         }
18     }
19     Other = {
20         __return = 0
21         block = {
22           "number": 0,
23           "timestamp": 0
24         }
25     }
26     Address_Map = [
27       {
28         "key": "ALL_OTHERS",
29         "value": "EmptyAddress"
30       }
31     ]
32
33 Function invocation is reverted.
```

Formal Verification Request 6

SafeMath sub

02, Jul 2019

⏱ 0.96 ms

Line 41-47 in File TaxaToken.sol

```
41    /*@CTK "SafeMath sub"
42      @post (a < b) == __reverted
43      @post !__reverted -> __return == a - b
44      @post !__reverted -> !__has_overflow
45      @post !__reverted -> !(__has_assertion_failure)
46      @post !(__has_buf_overflow)
47    */
```

Line 48-51 in File TaxaToken.sol

```
48    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
49        assert(b <= a);
50        return a - b;
51    }
```

✅ The code meets the specification.

Formal Verification Request 7

Method will not encounter an assertion failure.

📅 02, Jul 2019
⏱ 11.12 ms

Line 53 in File TaxaToken.sol

```
53    //@CTK FAIL NO_ASF
```

Line 61-65 in File TaxaToken.sol

```
61    function add(uint256 a, uint256 b) internal pure returns (uint256) {
62        uint256 c = a + b;
63        assert(c >= a);
64        return c;
65    }
```

❌ This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          a = 13
5          b = 243
6      }
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14           "gas": 0,
15           "sender": 0,
16           "value": 0
17         }
```

```
    }
    Other = {
        __return = 0
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 8

SafeMath add

02, Jul 2019
2.39 ms

Line 54-60 in File TaxaToken.sol

```
    /*@CTK "SafeMath add"
      @post (a + b < a || a + b < b) == __reverted
      @post !__reverted -> __return == a + b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
    */
```

Line 61-65 in File TaxaToken.sol

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

The code meets the specification.

Formal Verification Request 9

If method completes, integer overflow would not happen.

02, Jul 2019
72.83 ms

Line 108 in File TaxaToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 119-129 in File TaxaToken.sol

```
119     function transfer(address _to, uint256 _value) public returns (bool) {
120         require(_to != address(0));
121         require(_to != address(this));
122         require(_value <= balances[msg.sender]);
123 
124         // SafeMath.sub will throw if there is not enough balance.
125         balances[msg.sender] = balances[msg.sender].sub(_value);
126         balances[_to] = balances[_to].add(_value);
127         Transfer(msg.sender, _to, _value);
128         return true;
129     }
```

The code meets the specification.

Formal Verification Request 10

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
18.83 ms

Line 109 in File TaxaToken.sol

```
109     //@CTK NO_BUF_OVERFLOW
```

Line 119-129 in File TaxaToken.sol

```
119     function transfer(address _to, uint256 _value) public returns (bool) {
120         require(_to != address(0));
121         require(_to != address(this));
122         require(_value <= balances[msg.sender]);
123 
124         // SafeMath.sub will throw if there is not enough balance.
125         balances[msg.sender] = balances[msg.sender].sub(_value);
126         balances[_to] = balances[_to].add(_value);
127         Transfer(msg.sender, _to, _value);
128         return true;
129     }
```

The code meets the specification.

Formal Verification Request 11

Method will not encounter an assertion failure.

02, Jul 2019
30.76 ms

Line 110 in File TaxaToken.sol

```
110     //@CTK FAIL NO_ASF
```

Line 119-129 in File TaxaToken.sol

```
119     function transfer(address _to, uint256 _value) public returns (bool) {
120         require(_to != address(0));
121         require(_to != address(this));
```

```
122         require(_value <= balances[msg.sender]);
123
124         // SafeMath.sub will throw if there is not enough balance.
125         balances[msg.sender] = balances[msg.sender].sub(_value);
126         balances[_to] = balances[_to].add(_value);
127         Transfer(msg.sender, _to, _value);
128         return true;
129     }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          _to = 32
5          _value = 130
6      }
7      This = 0
8      Internal = {
9          __has_assertion_failure = false
10         __has_buf_overflow = false
11         __has_overflow = false
12         __has_returned = false
13         __reverted = false
14         msg = {
15           "gas": 0,
16           "sender": 0,
17           "value": 0
18         }
19     }
20     Other = {
21         __return = false
22         block = {
23           "number": 0,
24           "timestamp": 0
25         }
26     }
27     Address_Map = [
28       {
29         "key": 0,
30         "value": {
31           "contract_name": "BasicToken",
32           "balance": 0,
33           "contract": {
34             "balances": [
35               {
36                 "key": 0,
37                 "value": 192
38               },
39               {
40                 "key": 32,
41                 "value": 128
42               },
43               {
44                 "key": "ALL_OTHERS",
45                 "value": 130
46               }
47             ],
48             "totalSupply": 0
```

```
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 12

transfer

02, Jul 2019
106.82 ms

Line 111-118 in File TaxaToken.sol

```
    /*@CTK transfer
      @tag assume_completion
      @pre msg.sender != _to
      @post _to != address(0)
      @post _value <= balances[msg.sender]
      @post __post.balances[msg.sender] == balances[msg.sender] - _value
      @post __post.balances[_to] == balances[_to] + _value
     */
```

Line 119-129 in File TaxaToken.sol

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

        // SafeMath.sub will throw if there is not enough balance.
        balances[msg.sender] = balances[msg.sender].sub(_value);
        balances[_to] = balances[_to].add(_value);
        Transfer(msg.sender, _to, _value);
        return true;
    }
```

The code meets the specification.

Formal Verification Request 13

If method completes, integer overflow would not happen.

02, Jul 2019
5.25 ms

Line 136 in File TaxaToken.sol

```
    //@CTK NO_OVERFLOW
```

Line 142-144 in File TaxaToken.sol

```
142     function balanceOf(address _owner) public view returns (uint256 balance) {
143         return balances[_owner];
144     }
```

The code meets the specification.

Formal Verification Request 14

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
0.33 ms

Line 137 in File TaxaToken.sol

```
137     //@CTK NO_BUF_OVERFLOW
```

Line 142-144 in File TaxaToken.sol

```
142     function balanceOf(address _owner) public view returns (uint256 balance) {
143         return balances[_owner];
144     }
```

The code meets the specification.

Formal Verification Request 15

Method will not encounter an assertion failure.

02, Jul 2019
0.3 ms

Line 138 in File TaxaToken.sol

```
138     //@CTK NO_ASF
```

Line 142-144 in File TaxaToken.sol

```
142     function balanceOf(address _owner) public view returns (uint256 balance) {
143         return balances[_owner];
144     }
```

The code meets the specification.

Formal Verification Request 16

balanceOf

02, Jul 2019
0.3 ms

Line 139-141 in File TaxaToken.sol

```
139     /*@CTK balanceOf
140       @post balance == __post.balances[_owner]
141      */
```

Line 142-144 in File TaxaToken.sol

```
142     function balanceOf(address _owner) public view returns (uint256 balance) {
143         return balances[_owner];
144     }
```

The code meets the specification.

Formal Verification Request 17

If method completes, integer overflow would not happen.

02, Jul 2019
112.11 ms

Line 166 in File TaxaToken.sol

```
166     //@CTK NO_OVERFLOW
```

Line 178-189 in File TaxaToken.sol

```
178     function transferFrom(address _from, address _to, uint256 _value) public returns (
            bool) {
179         require(_to != address(0));
180         require(_to != address(this));
181         require(_value <= balances[_from]);
182         require(_value <= allowed[_from][msg.sender]);
183 
184         balances[_from] = balances[_from].sub(_value);
185         balances[_to] = balances[_to].add(_value);
186         allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
187         Transfer(_from, _to, _value);
188         return true;
189     }
```

The code meets the specification.

Formal Verification Request 18

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
26.56 ms

Line 167 in File TaxaToken.sol

```
167     //@CTK NO_BUF_OVERFLOW
```

Line 178-189 in File TaxaToken.sol

```
178     function transferFrom(address _from, address _to, uint256 _value) public returns (
            bool) {
179         require(_to != address(0));
180         require(_to != address(this));
181         require(_value <= balances[_from]);
182         require(_value <= allowed[_from][msg.sender]);
183 
```

```
184         balances[_from] = balances[_from].sub(_value);
185         balances[_to] = balances[_to].add(_value);
186         allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
187         Transfer(_from, _to, _value);
188         return true;
189     }
```

The code meets the specification.

Formal Verification Request 19

Method will not encounter an assertion failure.

02, Jul 2019
124.28 ms

Line 168 in File TaxaToken.sol

```
168     //@CTK FAIL NO_ASF
```

Line 178-189 in File TaxaToken.sol

```
178     function transferFrom(address _from, address _to, uint256 _value) public returns (
            bool) {
179         require(_to != address(0));
180         require(_to != address(this));
181         require(_value <= balances[_from]);
182         require(_value <= allowed[_from][msg.sender]);
183 
184         balances[_from] = balances[_from].sub(_value);
185         balances[_to] = balances[_to].add(_value);
186         allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
187         Transfer(_from, _to, _value);
188         return true;
189     }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          _from = 0
5          _to = 8
6          _value = 1
7      }
8      This = 0
9      Internal = {
10         __has_assertion_failure = false
11         __has_buf_overflow = false
12         __has_overflow = false
13         __has_returned = false
14         __reverted = false
15         msg = {
16           "gas": 0,
17           "sender": 0,
18           "value": 0
19         }
20     }
```

```
Other = {
    __return = false
    block = {
      "number": 0,
      "timestamp": 0
    }
}
Address_Map = [
  {
    "key": 0,
    "value": {
      "contract_name": "StandardToken",
      "balance": 0,
      "contract": {
        "allowed": [
          {
            "key": 128,
            "value": [
              {
                "key": 0,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 64
              }
            ]
          },
          {
            "key": 0,
            "value": [
              {
                "key": 0,
                "value": 128
              },
              {
                "key": 1,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 140
              }
            ]
          },
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": 140
              }
            ]
          }
        ],
        "balances": [
          {
            "key": 2,
```

```
              "value": 0
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": 0,
              "value": 141
            },
            {
              "key": 8,
              "value": 255
            },
            {
              "key": "ALL_OTHERS",
              "value": 140
            }
          ],
          "totalSupply": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 20

transferFrom correctness

02, Jul 2019
374.82 ms

Line 169-177 in File TaxaToken.sol

```
  /*@CTK "transferFrom correctness"
    @tag assume_completion
    @post _to != 0x0
    @post _value <= balances[_from] && _value <= allowed[_from][msg.sender]
    @post _to != _from -> __post.balances[_from] == balances[_from] - _value
    @post _to != _from -> __post.balances[_to] == balances[_to] + _value
    @post _to == _from -> __post.balances[_from] == balances[_from]
    @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
   */
```

Line 178-189 in File TaxaToken.sol

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

```
183
184         balances[_from] = balances[_from].sub(_value);
185         balances[_to] = balances[_to].add(_value);
186         allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
187         Transfer(_from, _to, _value);
188         return true;
189     }
```

The code meets the specification.

Formal Verification Request 21

If method completes, integer overflow would not happen.

02, Jul 2019
7.19 ms

Line 201 in File TaxaToken.sol

```
201     //@CTK NO_OVERFLOW
```

Line 207-211 in File TaxaToken.sol

```
207     function approve(address _spender, uint256 _value) public returns (bool) {
208         allowed[msg.sender][_spender] = _value;
209         Approval(msg.sender, _spender, _value);
210         return true;
211     }
```

The code meets the specification.

Formal Verification Request 22

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
0.34 ms

Line 202 in File TaxaToken.sol

```
202     //@CTK NO_BUF_OVERFLOW
```

Line 207-211 in File TaxaToken.sol

```
207     function approve(address _spender, uint256 _value) public returns (bool) {
208         allowed[msg.sender][_spender] = _value;
209         Approval(msg.sender, _spender, _value);
210         return true;
211     }
```

The code meets the specification.

Formal Verification Request 23

Method will not encounter an assertion failure.

02, Jul 2019
0.4 ms

Line 203 in File TaxaToken.sol

```
203     //@CTK NO_ASF
```

Line 207-211 in File TaxaToken.sol

```
207     function approve(address _spender, uint256 _value) public returns (bool) {
208         allowed[msg.sender][_spender] = _value;
209         Approval(msg.sender, _spender, _value);
210         return true;
211     }
```

The code meets the specification.

Formal Verification Request 24

approve correctness

02, Jul 2019
1.29 ms

Line 204-206 in File TaxaToken.sol

```
204     /*@CTK "approve correctness"
205       @post __post.allowed[msg.sender][_spender] == _value
206      */
```

Line 207-211 in File TaxaToken.sol

```
207     function approve(address _spender, uint256 _value) public returns (bool) {
208         allowed[msg.sender][_spender] = _value;
209         Approval(msg.sender, _spender, _value);
210         return true;
211     }
```

The code meets the specification.

Formal Verification Request 25

If method completes, integer overflow would not happen.

02, Jul 2019
4.49 ms

Line 219 in File TaxaToken.sol

```
219     //@CTK NO_OVERFLOW
```

Line 225-227 in File TaxaToken.sol

```
225     function allowance(address _owner, address _spender) public view returns (uint256)
            {
226         return allowed[_owner][_spender];
227     }
```

The code meets the specification.

Formal Verification Request 26

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
0.41 ms

Line 220 in File TaxaToken.sol

```
220     //@CTK NO_BUF_OVERFLOW
```

Line 225-227 in File TaxaToken.sol

```
225     function allowance(address _owner, address _spender) public view returns (uint256)
            {
226         return allowed[_owner][_spender];
227     }
```

The code meets the specification.

Formal Verification Request 27

Method will not encounter an assertion failure.

02, Jul 2019
0.32 ms

Line 221 in File TaxaToken.sol

```
221     //@CTK NO_ASF
```

Line 225-227 in File TaxaToken.sol

```
225     function allowance(address _owner, address _spender) public view returns (uint256)
            {
226         return allowed[_owner][_spender];
227     }
```

The code meets the specification.

Formal Verification Request 28

allowance correctness

02, Jul 2019
0.31 ms

Line 222-224 in File TaxaToken.sol

```
222     /*@CTK "allowance correctness"
223       @post __return == allowed[_owner][_spender]
224      */
```

Line 225-227 in File TaxaToken.sol

```
225     function allowance(address _owner, address _spender) public view returns (uint256)
            {
226         return allowed[_owner][_spender];
227     }
```

The code meets the specification.

Formal Verification Request 29

If method completes, integer overflow would not happen.

02, Jul 2019
29.37 ms

Line 239 in File TaxaToken.sol

```
239     //@CTK NO_OVERFLOW
```

Line 246-250 in File TaxaToken.sol

```
246     function increaseApproval(address _spender, uint256 _addedValue) public returns (
            bool){
247         allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
248         Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
249         return true;
250     }
```

The code meets the specification.

Formal Verification Request 30

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
0.7 ms

Line 240 in File TaxaToken.sol

```
240     //@CTK NO_BUF_OVERFLOW
```

Line 246-250 in File TaxaToken.sol

```
246     function increaseApproval(address _spender, uint256 _addedValue) public returns (
            bool){
247         allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
248         Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
249         return true;
250     }
```

The code meets the specification.

Formal Verification Request 31

Method will not encounter an assertion failure.

02, Jul 2019
5.17 ms

Line 241 in File TaxaToken.sol

```
241     //@CTK FAIL NO_ASF
```

Line 246-250 in File TaxaToken.sol

```
246     function increaseApproval(address _spender, uint256 _addedValue) public returns (
            bool){
247         allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
248         Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
249         return true;
250     }
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        _addedValue = 241
        _spender = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "StandardToken",
          "balance": 0,
          "contract": {
            "allowed": [
              {
                "key": 0,
                "value": [
                  {
```

```
                  "key": 4,
                  "value": 64
                },
                {
                  "key": 0,
                  "value": 143
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 241
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 241
                }
              ]
            }
          ],
          "balances": [
            {
              "key": 4,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 241
            }
          ],
          "totalSupply": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```


Formal Verification Request 32

increaseApproval correctness

🗓 02, Jul 2019
⏱ 1.7 ms

Line 242-245 in File TaxaToken.sol

```
  /*@CTK "increaseApproval correctness"
    @tag assume_completion
```

```
244      @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
             _addedValue
245      */
```

Line 246-250 in File TaxaToken.sol

```
246     function increaseApproval(address _spender, uint256 _addedValue) public returns (
            bool){
247         allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
248         Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
249         return true;
250     }
```

The code meets the specification.

Formal Verification Request 33

If method completes, integer overflow would not happen.

02, Jul 2019
34.27 ms

Line 262 in File TaxaToken.sol

```
262     //@CTK NO_OVERFLOW
```

Line 276-286 in File TaxaToken.sol

```
276     function decreaseApproval(address _spender, uint256 _subtractedValue) public
            returns (bool){
277         uint256 oldValue = allowed[msg.sender][_spender];
278         if (_subtractedValue > oldValue) {
279             allowed[msg.sender][_spender] = 0;
280         } else {
281             allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
282         }
283
284         Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
285         return true;
286     }
```

The code meets the specification.

Formal Verification Request 34

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
0.65 ms

Line 263 in File TaxaToken.sol

```
263     //@CTK NO_BUF_OVERFLOW
```

Line 276-286 in File TaxaToken.sol

```
276     function decreaseApproval(address _spender, uint256 _subtractedValue) public
            returns (bool){
277         uint256 oldValue = allowed[msg.sender][_spender];
278         if (_subtractedValue > oldValue) {
279             allowed[msg.sender][_spender] = 0;
280         } else {
281             allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
282         }
283
284         Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
285         return true;
286     }
```

The code meets the specification.

Formal Verification Request 35

Method will not encounter an assertion failure.

02, Jul 2019
1.08 ms

Line 264 in File TaxaToken.sol

```
264     //@CTK NO_ASF
```

Line 276-286 in File TaxaToken.sol

```
276     function decreaseApproval(address _spender, uint256 _subtractedValue) public
            returns (bool){
277         uint256 oldValue = allowed[msg.sender][_spender];
278         if (_subtractedValue > oldValue) {
279             allowed[msg.sender][_spender] = 0;
280         } else {
281             allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
282         }
283
284         Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
285         return true;
286     }
```

The code meets the specification.

Formal Verification Request 36

decreaseApproval0

02, Jul 2019
17.07 ms

Line 265-269 in File TaxaToken.sol

```
265     /*@CTK decreaseApproval0
266       @pre __return == true
267       @pre allowed[msg.sender][_spender] <= _subtractedValue
268       @post __post.allowed[msg.sender][_spender] == 0
269      */
```

Line 276-286 in File TaxaToken.sol

```
276     function decreaseApproval(address _spender, uint256 _subtractedValue) public
            returns (bool){
277         uint256 oldValue = allowed[msg.sender][_spender];
278         if (_subtractedValue > oldValue) {
279             allowed[msg.sender][_spender] = 0;
280         } else {
281             allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
282         }
283
284         Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
285         return true;
286     }
```

The code meets the specification.

Formal Verification Request 37

decreaseApproval

02, Jul 2019
10.55 ms

Line 270-275 in File TaxaToken.sol

```
270     /*@CTK decreaseApproval
271       @pre __return == true
272       @pre allowed[msg.sender][_spender] > _subtractedValue
273       @post __post.allowed[msg.sender][_spender] ==
274           allowed[msg.sender][_spender] - _subtractedValue
275      */
```

Line 276-286 in File TaxaToken.sol

```
276     function decreaseApproval(address _spender, uint256 _subtractedValue) public
            returns (bool){
277         uint256 oldValue = allowed[msg.sender][_spender];
278         if (_subtractedValue > oldValue) {
279             allowed[msg.sender][_spender] = 0;
280         } else {
281             allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
282         }
283
284         Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
285         return true;
286     }
```

The code meets the specification.

Formal Verification Request 38

transferOwnership

02, Jul 2019
18.26 ms

Line 320-324 in File TaxaToken.sol

```
320   /*@CTK transferOwnership
321     @tag assume_completion
322     @post newOwner != address(0)
323     @post __post.owner == newOwner
324    */
```

Line 325-329 in File TaxaToken.sol

```
325   function transferOwnership(address newOwner) public onlyOwner {
326       require(newOwner != address(0));
327       OwnershipTransferred(owner, newOwner);
328       owner = newOwner;
329   }
```

✅ The code meets the specification.

Formal Verification Request 39

pause

📅 02, Jul 2019
⏱ 19.15 ms

Line 364-369 in File TaxaToken.sol

```
364   /*@CTK pause
365     @tag assume_completion
366     @post paused == false
367     @post owner == msg.sender
368     @post __post.paused == true
369    */
```

Line 370-373 in File TaxaToken.sol

```
370   function pause() onlyOwner whenNotPaused public {
371       paused = true;
372       Pause();
373   }
```

✅ The code meets the specification.

Formal Verification Request 40

unpause

📅 02, Jul 2019
⏱ 18.5 ms

Line 378-383 in File TaxaToken.sol

```
378   /*@CTK unpause
379     @tag assume_completion
380     @post paused == true
381     @post owner == msg.sender
382     @post __post.paused == false
383    */
```

Line 384-387 in File TaxaToken.sol

```
384     function unpause() onlyOwner whenPaused public {
385         paused = false;
386         Unpause();
387     }
```

The code meets the specification.

Formal Verification Request 41

transfer

02, Jul 2019
245.66 ms

Line 398-406 in File TaxaToken.sol

```
398     /*@CTK transfer
399       @tag assume_completion
400       @pre msg.sender != _to
401       @post !paused
402       @post _to != address(0)
403       @post _value <= balances[msg.sender]
404       @post __post.balances[msg.sender] == balances[msg.sender] - _value
405       @post __post.balances[_to] == balances[_to] + _value
406      */
```

Line 407-409 in File TaxaToken.sol

```
407     function transfer(address _to, uint256 _value) public whenNotPaused returns (bool)
            {
408         return super.transfer(_to, _value);
409     }
```

The code meets the specification.

Formal Verification Request 42

If method completes, integer overflow would not happen.

02, Jul 2019
215.39 ms

Line 411 in File TaxaToken.sol

```
411     //@CTK NO_OVERFLOW
```

Line 424-426 in File TaxaToken.sol

```
424     function transferFrom(address _from, address _to, uint256 _value) public
            whenNotPaused returns (bool) {
425         return super.transferFrom(_from, _to, _value);
426     }
```

The code meets the specification.

Formal Verification Request 43

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
20.24 ms

Line 412 in File TaxaToken.sol

```
412     //@CTK NO_BUF_OVERFLOW
```

Line 424-426 in File TaxaToken.sol

```
424     function transferFrom(address _from, address _to, uint256 _value) public
            whenNotPaused returns (bool) {
425         return super.transferFrom(_from, _to, _value);
426     }
```

The code meets the specification.

Formal Verification Request 44

Method will not encounter an assertion failure.

02, Jul 2019
156.82 ms

Line 413 in File TaxaToken.sol

```
413     //@CTK FAIL NO_ASF
```

Line 424-426 in File TaxaToken.sol

```
424     function transferFrom(address _from, address _to, uint256 _value) public
            whenNotPaused returns (bool) {
425         return super.transferFrom(_from, _to, _value);
426     }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          _from = 0
5          _to = 4
6          _value = 112
7      }
8      This = 0
9      Internal = {
10         __has_assertion_failure = false
11         __has_buf_overflow = false
12         __has_overflow = false
13         __has_returned = false
14         __reverted = false
15         msg = {
16           "gas": 0,
17           "sender": 0,
18           "value": 0
19         }
```

```
}
Other = {
    __return = false
    block = {
      "number": 0,
      "timestamp": 0
    }
}
Address_Map = [
  {
    "key": 0,
    "value": {
      "contract_name": "PausableToken",
      "balance": 0,
      "contract": {
        "paused": false,
        "owner": 0,
        "allowed": [
          {
            "key": 0,
            "value": [
              {
                "key": 0,
                "value": 128
              },
              {
                "key": 16,
                "value": 2
              },
              {
                "key": 2,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 112
              }
            ]
          },
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": 112
              }
            ]
          }
        ],
        "balances": [
          {
            "key": 36,
            "value": 0
          },
          {
            "key": 4,
            "value": 174
          },
```

```
            {
              "key": 0,
              "value": 128
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": 128,
              "value": 0
            },
            {
              "key": 64,
              "value": 0
            },
            {
              "key": 1,
              "value": 0
            },
            {
              "key": 132,
              "value": 32
            },
            {
              "key": 32,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 112
            }
          ],
          "totalSupply": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 45

transferFrom correctness

🗓 02, Jul 2019
⏱ 413.74 ms

Line 414-423 in File TaxaToken.sol

```
  /*@CTK "transferFrom correctness"
    @tag assume_completion
    @post !paused
```

```
417      @post _to != 0x0
418      @post _value <= balances[_from] && _value <= allowed[_from][msg.sender]
419      @post _to != _from -> __post.balances[_from] == balances[_from] - _value
420      @post _to != _from -> __post.balances[_to] == balances[_to] + _value
421      @post _to == _from -> __post.balances[_from] == balances[_from]
422      @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
423     */
```

Line 424-426 in File TaxaToken.sol

```
424     function transferFrom(address _from, address _to, uint256 _value) public
            whenNotPaused returns (bool) {
425         return super.transferFrom(_from, _to, _value);
426     }
```

The code meets the specification.

Formal Verification Request 46

If method completes, integer overflow would not happen.

02, Jul 2019
34.18 ms

Line 428 in File TaxaToken.sol

```
428     //@CTK NO_OVERFLOW
```

Line 435-437 in File TaxaToken.sol

```
435     function approve(address _spender, uint256 _value) public whenNotPaused returns (
            bool) {
436         return super.approve(_spender, _value);
437     }
```

The code meets the specification.

Formal Verification Request 47

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
1.26 ms

Line 429 in File TaxaToken.sol

```
429     //@CTK NO_BUF_OVERFLOW
```

Line 435-437 in File TaxaToken.sol

```
435     function approve(address _spender, uint256 _value) public whenNotPaused returns (
            bool) {
436         return super.approve(_spender, _value);
437     }
```

The code meets the specification.

Formal Verification Request 48

Method will not encounter an assertion failure.

📅 02, Jul 2019
⏱ 0.75 ms

Line 430 in File TaxaToken.sol

```
430     //@CTK NO_ASF
```

Line 435-437 in File TaxaToken.sol

```
435     function approve(address _spender, uint256 _value) public whenNotPaused returns (
            bool) {
436         return super.approve(_spender, _value);
437     }
```

✅ The code meets the specification.

Formal Verification Request 49

approve correctness

📅 02, Jul 2019
⏱ 2.34 ms

Line 431-434 in File TaxaToken.sol

```
431     /*@CTK "approve correctness"
432       @post paused -> __reverted
433       @post !paused -> __post.allowed[msg.sender][_spender] == _value
434      */
```

Line 435-437 in File TaxaToken.sol

```
435     function approve(address _spender, uint256 _value) public whenNotPaused returns (
            bool) {
436         return super.approve(_spender, _value);
437     }
```

✅ The code meets the specification.

Formal Verification Request 50

If method completes, integer overflow would not happen.

📅 02, Jul 2019
⏱ 75.77 ms

Line 439 in File TaxaToken.sol

```
439     //@CTK NO_OVERFLOW
```

Line 447-449 in File TaxaToken.sol

```
447     function increaseApproval(address _spender, uint256 _addedValue) public
            whenNotPaused returns (bool){
448         return super.increaseApproval(_spender, _addedValue);
449     }
```

The code meets the specification.

Formal Verification Request 51

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
0.91 ms

Line 440 in File TaxaToken.sol

```
440     //@CTK NO_BUF_OVERFLOW
```

Line 447-449 in File TaxaToken.sol

```
447     function increaseApproval(address _spender, uint256 _addedValue) public
            whenNotPaused returns (bool){
448         return super.increaseApproval(_spender, _addedValue);
449     }
```

The code meets the specification.

Formal Verification Request 52

Method will not encounter an assertion failure.

02, Jul 2019
7.74 ms

Line 441 in File TaxaToken.sol

```
441     //@CTK FAIL NO_ASF
```

Line 447-449 in File TaxaToken.sol

```
447     function increaseApproval(address _spender, uint256 _addedValue) public
            whenNotPaused returns (bool){
448         return super.increaseApproval(_spender, _addedValue);
449     }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          _addedValue = 172
5          _spender = 0
6      }
7      This = 0
8      Internal = {
9          __has_assertion_failure = false
10         __has_buf_overflow = false
11         __has_overflow = false
```

```
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "PausableToken",
          "balance": 0,
          "contract": {
            "paused": false,
            "owner": 0,
            "allowed": [
              {
                "key": 0,
                "value": [
                  {
                    "key": 0,
                    "value": 208
                  },
                  {
                    "key": 8,
                    "value": 0
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 172
                  }
                ]
              },
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 124
                  }
                ]
              }
            ],
            "balances": [
              {
                "key": 8,
                "value": 1
              },
              {
```

```
              "key": "ALL_OTHERS",
              "value": 172
            }
          ],
          "totalSupply": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 53

increaseApproval correctness

02, Jul 2019
8.49 ms

Line 442-446 in File TaxaToken.sol

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

Line 447-449 in File TaxaToken.sol

```
  function increaseApproval(address _spender, uint256 _addedValue) public
      whenNotPaused returns (bool){
      return super.increaseApproval(_spender, _addedValue);
  }
```

The code meets the specification.

Formal Verification Request 54

If method completes, integer overflow would not happen.

02, Jul 2019
89.56 ms

Line 451 in File TaxaToken.sol

```
  //@CTK NO_OVERFLOW
```

Line 467-469 in File TaxaToken.sol

```
  function decreaseApproval(address _spender, uint256 _subtractedValue) public
      whenNotPaused returns (bool){
      return super.decreaseApproval(_spender, _subtractedValue);
  }
```

The code meets the specification.

Formal Verification Request 55

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
1.07 ms

Line 452 in File TaxaToken.sol

```
452     //@CTK NO_BUF_OVERFLOW
```

Line 467-469 in File TaxaToken.sol

```
467     function decreaseApproval(address _spender, uint256 _subtractedValue) public
            whenNotPaused returns (bool){
468         return super.decreaseApproval(_spender, _subtractedValue);
469     }
```

The code meets the specification.

Formal Verification Request 56

Method will not encounter an assertion failure.

02, Jul 2019
1.55 ms

Line 453 in File TaxaToken.sol

```
453     //@CTK NO_ASF
```

Line 467-469 in File TaxaToken.sol

```
467     function decreaseApproval(address _spender, uint256 _subtractedValue) public
            whenNotPaused returns (bool){
468         return super.decreaseApproval(_spender, _subtractedValue);
469     }
```

The code meets the specification.

Formal Verification Request 57

decreaseApproval0

02, Jul 2019
22.65 ms

Line 454-459 in File TaxaToken.sol

```
454     /*@CTK decreaseApproval0
455       @pre __return == true
456       @pre !paused
457       @pre allowed[msg.sender][_spender] <= _subtractedValue
458       @post __post.allowed[msg.sender][_spender] == 0
459      */
```

Line 467-469 in File TaxaToken.sol

```
467     function decreaseApproval(address _spender, uint256 _subtractedValue) public
            whenNotPaused returns (bool){
468         return super.decreaseApproval(_spender, _subtractedValue);
469     }
```

The code meets the specification.

Formal Verification Request 58

decreaseApproval

02, Jul 2019
16.22 ms

Line 460-466 in File TaxaToken.sol

```
460     /*@CTK decreaseApproval
461       @pre __return == true
462       @pre !paused
463       @pre allowed[msg.sender][_spender] > _subtractedValue
464       @post __post.allowed[msg.sender][_spender] ==
465           allowed[msg.sender][_spender] - _subtractedValue
466      */
```

Line 467-469 in File TaxaToken.sol

```
467     function decreaseApproval(address _spender, uint256 _subtractedValue) public
            whenNotPaused returns (bool){
468         return super.decreaseApproval(_spender, _subtractedValue);
469     }
```

The code meets the specification.

Formal Verification Request 59

If method completes, integer overflow would not happen.

02, Jul 2019
7.35 ms

Line 486 in File TaxaToken.sol

```
486     //@CTK NO_OVERFLOW
```

Line 489-492 in File TaxaToken.sol

```
489     function TaxaNetworkToken() public {
490         balances[owner] = totalSupply;
491         Transfer(address(0), owner, balances[owner]);
492     }
```

The code meets the specification.

Formal Verification Request 60

Buffer overflow / array index out of bound would never happen.

02, Jul 2019
0.28 ms

Line 487 in File TaxaToken.sol

```
487     //@CTK NO_BUF_OVERFLOW
```

Line 489-492 in File TaxaToken.sol

```
489     function TaxaNetworkToken() public {
490         balances[owner] = totalSupply;
491         Transfer(address(0), owner, balances[owner]);
492     }
```

The code meets the specification.

Formal Verification Request 61

Method will not encounter an assertion failure.

02, Jul 2019
0.27 ms

Line 488 in File TaxaToken.sol

```
488     //@CTK NO_ASF
```

Line 489-492 in File TaxaToken.sol

```
489     function TaxaNetworkToken() public {
490         balances[owner] = totalSupply;
491         Transfer(address(0), owner, balances[owner]);
492     }
```

The code meets the specification.

Formal Verification Request 62

Method will not encounter an assertion failure.

02, Jul 2019
18.24 ms

Line 8 in File TaxaLockFoundation.sol

```
8     //@CTK FAIL NO_ASF
```

Line 16-23 in File TaxaLockFoundation.sol

```
16    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
17        if (a == 0) {
18            return 0;
19        }
20        uint256 c = a * b;
```

```
21          assert(c / a == b);
22          return c;
23      }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          a = 2
5          b = 156
6      }
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14           "gas": 0,
15           "sender": 0,
16           "value": 0
17         }
18     }
19     Other = {
20         __return = 0
21         block = {
22           "number": 0,
23           "timestamp": 0
24         }
25     }
26     Address_Map = [
27       {
28         "key": "ALL_OTHERS",
29         "value": "EmptyAddress"
30       }
31     ]
32
33 Function invocation is reverted.
```

Formal Verification Request 63

SafeMath mul

02, Jul 2019
297.08 ms

Line 9-15 in File TaxaLockFoundation.sol

```
9       /*@CTK "SafeMath mul"
10        @post ((a > 0) && (((a * b) / a) != b)) == (__reverted)
11        @post !__reverted -> __return == a * b
12        @post !__reverted == !__has_overflow
13        @post !__reverted -> !(__has_assertion_failure)
14        @post !(__has_buf_overflow)
15      */
```

Line 16-23 in File TaxaLockFoundation.sol

```
16      function mul(uint256 a, uint256 b) internal pure returns (uint256) {
17          if (a == 0) {
18              return 0;
19          }
20          uint256 c = a * b;
21          assert(c / a == b);
22          return c;
23      }
```

✅ The code meets the specification.

Formal Verification Request 64

Method will not encounter an assertion failure.

📅 02, Jul 2019
⏱ 6.31 ms

Line 25 in File TaxaLockFoundation.sol

```
25      //@CTK FAIL NO_ASF
```

Line 33-38 in File TaxaLockFoundation.sol

```
33      function div(uint256 a, uint256 b) internal pure returns (uint256) {
34          // assert(b > 0); // Solidity automatically throws when dividing by 0
35          uint256 c = a / b;
36          // assert(a == b * c + a % b); // There is no case in which this doesn't hold
37          return c;
38      }
```

❌ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        a = 0
        b = 0
    }
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = 0
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
```

```
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 65

SafeMath div

02, Jul 2019
0.98 ms

Line 26-32 in File TaxaLockFoundation.sol

```
    /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
    */
```

Line 33-38 in File TaxaLockFoundation.sol

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

The code meets the specification.

Formal Verification Request 66

Method will not encounter an assertion failure.

02, Jul 2019
11.8 ms

Line 40 in File TaxaLockFoundation.sol

```
    //@CTK FAIL NO_ASF
```

Line 48-51 in File TaxaLockFoundation.sol

```
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        a = 0
        b = 1
    }
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = 0
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 67

SafeMath sub

📅 02, Jul 2019
⏱ 0.79 ms

Line 41-47 in File TaxaLockFoundation.sol

```
    /*@CTK "SafeMath sub"
      @post (a < b) == __reverted
      @post !__reverted -> __return == a - b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
    */
```

Line 48-51 in File TaxaLockFoundation.sol

```
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }
```

The code meets the specification.

Formal Verification Request 68

Method will not encounter an assertion failure.

02, Jul 2019
12.04 ms

Line 53 in File TaxaLockFoundation.sol

```
53      //@CTK FAIL NO_ASF
```

Line 60-64 in File TaxaLockFoundation.sol

```
60      function add(uint256 a, uint256 b) internal pure returns (uint256) {
61          uint256 c = a + b;
62          assert(c >= a);
63          return c;
64      }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          a = 13
5          b = 243
6      }
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14           "gas": 0,
15           "sender": 0,
16           "value": 0
17         }
18     }
19     Other = {
20         __return = 0
21         block = {
22           "number": 0,
23           "timestamp": 0
24         }
25     }
26     Address_Map = [
27       {
28         "key": "ALL_OTHERS",
29         "value": "EmptyAddress"
30       }
31     ]
32
33 Function invocation is reverted.
```

Formal Verification Request 69

SafeMath add

📅 02, Jul 2019
⏱ 2.2 ms

Line 54-59 in File TaxaLockFoundation.sol

```
54    /*@CTK "SafeMath add"
55      @post (a + b < a || a + b < b) == __reverted
56      @post !__reverted -> __return == a + b
57      @post !__reverted -> !__has_overflow
58      @post !(__has_buf_overflow)
59    */
```

Line 60-64 in File TaxaLockFoundation.sol

```
60    function add(uint256 a, uint256 b) internal pure returns (uint256) {
61        uint256 c = a + b;
62        assert(c >= a);
63        return c;
64    }
```

✅ The code meets the specification.

Formal Verification Request 70

transferOwnership

📅 02, Jul 2019
⏱ 17.42 ms

Line 98-102 in File TaxaLockFoundation.sol

```
98    /*@CTK transferOwnership
99      @tag assume_completion
100     @post newOwner != address(0)
101     @post __post.owner == newOwner
102    */
```

Line 103-107 in File TaxaLockFoundation.sol

```
103   function transferOwnership(address newOwner) public onlyOwner {
104       require(newOwner != address(0));
105       emit OwnershipTransferred(owner, newOwner);
106       owner = newOwner;
107   }
```

✅ The code meets the specification.

Formal Verification Request 71

constructor

📅 02, Jul 2019
⏱ 60.03 ms

Line 207-216 in File TaxaLockFoundation.sol

```
207  /*@CTK "constructor"
208    @tag assume_completion
209    @post _beneficiary != address(0)
210    @post _cliff <= _duration
211    @post __post.beneficiary == _beneficiary
212    @post __post.revocable == _revocable
213    @post __post.duration == _duration
214    @post __post.start == _start
215    @post __post.cliff == _start + _cliff
216   */
```

Line 217-236 in File TaxaLockFoundation.sol

```
217  constructor(
218    ERC20Basic _token,
219    address _beneficiary,
220    uint256 _start,
221    uint256 _cliff,
222    uint256 _duration,
223    bool _revocable
224  )
225    public
226  {
227    require(_beneficiary != address(0));
228    require(_cliff <= _duration);
229
230    token = _token;
231    beneficiary = _beneficiary;
232    revocable = _revocable;
233    duration = _duration;
234    cliff = _start.add(_cliff);
235    start = _start;
236  }
```

The code meets the specification.

Formal Verification Request 72

vestedAmount_not_cliff_yet

02, Jul 2019
105.25 ms

Line 298-302 in File TaxaLockFoundation.sol

```
298  /*@CTK vestedAmount_not_cliff_yet
299    @tag assume_completion
300    @pre now < cliff
301    @post __return == 0
302   */
```

Line 313-324 in File TaxaLockFoundation.sol

```
313  function vestedAmount() public view returns (uint256) {
314    uint256 currentBalance = balances[token];
315    uint256 totalBalance = currentBalance.add(released);
```

```
316
317     if (block.timestamp < cliff) {
318       return 0;
319     } else if (block.timestamp >= start.add(duration) || revoked) {
320       return totalBalance;
321     } else {
322       return totalBalance.mul(block.timestamp.sub(start)).div(duration);
323     }
324   }
```

The code meets the specification.

Formal Verification Request 73

remainingAmountAreRevokedOrReleased

02, Jul 2019
28.11 ms

Line 303-307 in File TaxaLockFoundation.sol

```
303   /*@CTK remainingAmountAreRevokedOrReleased
304     @tag assume_completion
305     @pre (now >= start + duration || revoked) && (now >= cliff)
306     @post __return == balances[token] + released
307    */
```

Line 313-324 in File TaxaLockFoundation.sol

```
313   function vestedAmount() public view returns (uint256) {
314     uint256 currentBalance = balances[token];
315     uint256 totalBalance = currentBalance.add(released);
316
317     if (block.timestamp < cliff) {
318       return 0;
319     } else if (block.timestamp >= start.add(duration) || revoked) {
320       return totalBalance;
321     } else {
322       return totalBalance.mul(block.timestamp.sub(start)).div(duration);
323     }
324   }
```

The code meets the specification.

Source Code with CertiK Labels

File TaxaToken.sol

```
pragma solidity ^0.4.18;

/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
library SafeMath {
    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath mul"
      @post ((a > 0) && (((a * b) / a) != b)) == (__reverted)
      @post !__reverted -> __return == a * b
      @post !__reverted == !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
    */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }
        uint256 c = a * b;
        assert(c / a == b);
        return c;
    }

    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
    */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        // assert(b > 0); // Solidity automatically throws when dividing by 0
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold
        return c;
    }

    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath sub"
      @post (a < b) == __reverted
      @post !__reverted -> __return == a - b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !(__has_assertion_failure)
      @post !(__has_buf_overflow)
    */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }

    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath add"
```

```
55        @post (a + b < a || a + b < b) == __reverted
56        @post !__reverted -> __return == a + b
57        @post !__reverted -> !__has_overflow
58        @post !__reverted -> !(__has_assertion_failure)
59        @post !(__has_buf_overflow)
60    */
61    function add(uint256 a, uint256 b) internal pure returns (uint256) {
62        uint256 c = a + b;
63        assert(c >= a);
64        return c;
65    }
66 }
67
68
69 /**
70  * @title ERC20Basic
71  * @dev Simpler version of ERC20 interface
72  * @dev see https://github.com/ethereum/EIPs/issues/179
73  */
74 contract ERC20Basic {
75    uint256 public totalSupply;
76    function balanceOf(address who) public view returns (uint256);
77    function transfer(address to, uint256 value) public returns (bool);
78    event Transfer(address indexed from, address indexed to, uint256 value);
79 }
80
81
82 /**
83  * @title ERC20 interface
84  * @dev see https://github.com/ethereum/EIPs/issues/20
85  */
86 contract ERC20 is ERC20Basic {
87    function allowance(address owner, address spender) public view returns (uint256);
88    function transferFrom(address from, address to, uint256 value) public returns (
          bool);
89    function approve(address spender, uint256 value) public returns (bool);
90    event Approval(address indexed owner, address indexed spender, uint256 value);
91 }
92
93
94 /**
95  * @title Basic token
96  * @dev Basic version of StandardToken, with no allowances.
97  */
98 contract BasicToken is ERC20Basic {
99    using SafeMath for uint256;
100
101   mapping(address => uint256) balances;
102
103   /**
104   * @dev transfer token for a specified address
105   * @param _to The address to transfer to.
106   * @param _value The amount to be transferred.
107   */
108   //@CTK NO_OVERFLOW
109   //@CTK NO_BUF_OVERFLOW
110   //@CTK FAIL NO_ASF
111   /*@CTK transfer
```

```
112       @tag assume_completion
113       @pre msg.sender != _to
114       @post _to != address(0)
115       @post _value <= balances[msg.sender]
116       @post __post.balances[msg.sender] == balances[msg.sender] - _value
117       @post __post.balances[_to] == balances[_to] + _value
118      */
119     function transfer(address _to, uint256 _value) public returns (bool) {
120         require(_to != address(0));
121         require(_to != address(this));
122         require(_value <= balances[msg.sender]);
123 
124         // SafeMath.sub will throw if there is not enough balance.
125         balances[msg.sender] = balances[msg.sender].sub(_value);
126         balances[_to] = balances[_to].add(_value);
127         Transfer(msg.sender, _to, _value);
128         return true;
129     }
130 
131     /**
132     * @dev Gets the balance of the specified address.
133     * @param _owner The address to query the the balance of.
134     * @return An uint256 representing the amount owned by the passed address.
135     */
136     //@CTK NO_OVERFLOW
137     //@CTK NO_BUF_OVERFLOW
138     //@CTK NO_ASF
139     /*@CTK balanceOf
140       @post balance == __post.balances[_owner]
141      */
142     function balanceOf(address _owner) public view returns (uint256 balance) {
143         return balances[_owner];
144     }
145 
146 }
147 
148 
149 /**
150  * @title Standard ERC20 token
151  *
152  * @dev Implementation of the basic standard token.
153  * @dev https://github.com/ethereum/EIPs/issues/20
154  * @dev Based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master
       /smart_contract/FirstBloodToken.sol
155  */
156 contract StandardToken is ERC20, BasicToken {
157 
158     mapping (address => mapping (address => uint256)) internal allowed;
159 
160     /**
161      * @dev Transfer tokens from one address to another
162      * @param _from address The address which you want to send tokens from
163      * @param _to address The address which you want to transfer to
164      * @param _value uint256 the amount of tokens to be transferred
165      */
166     //@CTK NO_OVERFLOW
167     //@CTK NO_BUF_OVERFLOW
168     //@CTK FAIL NO_ASF
```

```
169     /*@CTK "transferFrom correctness"
170       @tag assume_completion
171       @post _to != 0x0
172       @post _value <= balances[_from] && _value <= allowed[_from][msg.sender]
173       @post _to != _from -> __post.balances[_from] == balances[_from] - _value
174       @post _to != _from -> __post.balances[_to] == balances[_to] + _value
175       @post _to == _from -> __post.balances[_from] == balances[_from]
176       @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
177      */
178     function transferFrom(address _from, address _to, uint256 _value) public returns (
            bool) {
179         require(_to != address(0));
180         require(_to != address(this));
181         require(_value <= balances[_from]);
182         require(_value <= allowed[_from][msg.sender]);
183 
184         balances[_from] = balances[_from].sub(_value);
185         balances[_to] = balances[_to].add(_value);
186         allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
187         Transfer(_from, _to, _value);
188         return true;
189     }
190 
191     /**
192      * @dev Approve the passed address to spend the specified amount of tokens on
            behalf of msg.sender.
193      *
194      * Beware that changing an allowance with this method brings the risk that someone
            may use both the old
195      * and the new allowance by unfortunate transaction ordering. One possible
            solution to mitigate this
196      * race condition is to first reduce the spender's allowance to 0 and set the
            desired value afterwards:
197      * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
198      * @param _spender The address which will spend the funds.
199      * @param _value The amount of tokens to be spent.
200      */
201     //@CTK NO_OVERFLOW
202     //@CTK NO_BUF_OVERFLOW
203     //@CTK NO_ASF
204     /*@CTK "approve correctness"
205       @post __post.allowed[msg.sender][_spender] == _value
206      */
207     function approve(address _spender, uint256 _value) public returns (bool) {
208         allowed[msg.sender][_spender] = _value;
209         Approval(msg.sender, _spender, _value);
210         return true;
211     }
212 
213     /**
214      * @dev Function to check the amount of tokens that an owner allowed to a spender.
215      * @param _owner address The address which owns the funds.
216      * @param _spender address The address which will spend the funds.
217      * @return An uint256 specifying the amount of tokens still available for the
            spender.
218      */
219     //@CTK NO_OVERFLOW
220     //@CTK NO_BUF_OVERFLOW
```

```
221     //@CTK NO_ASF
222     /*@CTK "allowance correctness"
223       @post __return == allowed[_owner][_spender]
224      */
225     function allowance(address _owner, address _spender) public view returns (uint256)
            {
226         return allowed[_owner][_spender];
227     }
228
229     /**
230      * @dev Increase the amount of tokens that an owner allowed to a spender.
231      *
232      * approve should be called when allowed[_spender] == 0. To increment
233      * allowed value is better to use this function to avoid 2 calls (and wait until
234      * the first transaction is mined)
235      * From MonolithDAO Token.sol
236      * @param _spender The address which will spend the funds.
237      * @param _addedValue The amount of tokens to increase the allowance by.
238      */
239     //@CTK NO_OVERFLOW
240     //@CTK NO_BUF_OVERFLOW
241     //@CTK FAIL NO_ASF
242     /*@CTK "increaseApproval correctness"
243       @tag assume_completion
244       @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
              _addedValue
245      */
246     function increaseApproval(address _spender, uint256 _addedValue) public returns (
            bool){
247         allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
248         Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
249         return true;
250     }
251
252     /**
253      * @dev Decrease the amount of tokens that an owner allowed to a spender.
254      *
255      * approve should be called when allowed[_spender] == 0. To decrement
256      * allowed value is better to use this function to avoid 2 calls (and wait until
257      * the first transaction is mined)
258      * From MonolithDAO Token.sol
259      * @param _spender The address which will spend the funds.
260      * @param _subtractedValue The amount of tokens to decrease the allowance by.
261      */
262     //@CTK NO_OVERFLOW
263     //@CTK NO_BUF_OVERFLOW
264     //@CTK NO_ASF
265     /*@CTK decreaseApproval0
266       @pre __return == true
267       @pre allowed[msg.sender][_spender] <= _subtractedValue
268       @post __post.allowed[msg.sender][_spender] == 0
269      */
270     /*@CTK decreaseApproval
271       @pre __return == true
272       @pre allowed[msg.sender][_spender] > _subtractedValue
273       @post __post.allowed[msg.sender][_spender] ==
274           allowed[msg.sender][_spender] - _subtractedValue
275      */
```

```
276     function decreaseApproval(address _spender, uint256 _subtractedValue) public
            returns (bool){
277         uint256 oldValue = allowed[msg.sender][_spender];
278         if (_subtractedValue > oldValue) {
279             allowed[msg.sender][_spender] = 0;
280         } else {
281             allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
282         }
283
284         Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
285         return true;
286     }
287 }
288
289
290 /**
291  * @title Ownable
292  * @dev The Ownable contract has an owner address, and provides basic authorization
        control
293  * functions, this simplifies the implementation of "user permissions".
294  */
295 contract Ownable {
296     address public owner;
297
298     event OwnershipTransferred(address indexed previousOwner, address indexed newOwner
            );
299
300     /**
301      * @dev The Ownable constructor sets the original `owner` of the contract to the
            sender
302      * account.
303      */
304     function Ownable() public {
305         owner = msg.sender;
306     }
307
308     /**
309      * @dev Throws if called by any account other than the owner.
310      */
311     modifier onlyOwner() {
312         require(msg.sender == owner);
313         _;
314     }
315
316     /**
317      * @dev Allows the current owner to transfer control of the contract to a newOwner
            .
318      * @param newOwner The address to transfer ownership to.
319      */
320     /*@CTK transferOwnership
321       @tag assume_completion
322       @post newOwner != address(0)
323       @post __post.owner == newOwner
324      */
325     function transferOwnership(address newOwner) public onlyOwner {
326         require(newOwner != address(0));
327         OwnershipTransferred(owner, newOwner);
328         owner = newOwner;
```

```
329     }
330
331 }
332
333
334 /**
335  * @title Pausable
336  * @dev Base contract which allows children to implement an emergency stop mechanism.
337  */
338 contract Pausable is Ownable {
339     event Pause();
340     event Unpause();
341
342     bool public paused = false;
343
344
345     /**
346      * @dev Modifier to make a function callable only when the contract is not paused.
347      */
348     modifier whenNotPaused() {
349         require(!paused);
350         _;
351     }
352
353     /**
354      * @dev Modifier to make a function callable only when the contract is paused.
355      */
356     modifier whenPaused() {
357         require(paused);
358         _;
359     }
360
361     /**
362      * @dev called by the owner to pause, triggers stopped state
363      */
364     /*@CTK pause
365       @tag assume_completion
366       @post paused == false
367       @post owner == msg.sender
368       @post __post.paused == true
369      */
370     function pause() onlyOwner whenNotPaused public {
371         paused = true;
372         Pause();
373     }
374
375     /**
376      * @dev called by the owner to unpause, returns to normal state
377      */
378     /*@CTK unpause
379       @tag assume_completion
380       @post paused == true
381       @post owner == msg.sender
382       @post __post.paused == false
383      */
384     function unpause() onlyOwner whenPaused public {
385         paused = false;
386         Unpause();
```

```
387     }
388 }
389
390
391 /**
392  * @title Pausable token
393  *
394  * @dev StandardToken modified with pausable transfers.
395  **/
396 contract PausableToken is StandardToken, Pausable {
397
398     /*@CTK transfer
399       @tag assume_completion
400       @pre msg.sender != _to
401       @post !paused
402       @post _to != address(0)
403       @post _value <= balances[msg.sender]
404       @post __post.balances[msg.sender] == balances[msg.sender] - _value
405       @post __post.balances[_to] == balances[_to] + _value
406      */
407     function transfer(address _to, uint256 _value) public whenNotPaused returns (bool)
          {
408         return super.transfer(_to, _value);
409     }
410
411     //@CTK NO_OVERFLOW
412     //@CTK NO_BUF_OVERFLOW
413     //@CTK FAIL NO_ASF
414     /*@CTK "transferFrom correctness"
415       @tag assume_completion
416       @post !paused
417       @post _to != 0x0
418       @post _value <= balances[_from] && _value <= allowed[_from][msg.sender]
419       @post _to != _from -> __post.balances[_from] == balances[_from] - _value
420       @post _to != _from -> __post.balances[_to] == balances[_to] + _value
421       @post _to == _from -> __post.balances[_from] == balances[_from]
422       @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
423      */
424     function transferFrom(address _from, address _to, uint256 _value) public
          whenNotPaused returns (bool) {
425         return super.transferFrom(_from, _to, _value);
426     }
427
428     //@CTK NO_OVERFLOW
429     //@CTK NO_BUF_OVERFLOW
430     //@CTK NO_ASF
431     /*@CTK "approve correctness"
432       @post paused -> __reverted
433       @post !paused -> __post.allowed[msg.sender][_spender] == _value
434      */
435     function approve(address _spender, uint256 _value) public whenNotPaused returns (
          bool) {
436         return super.approve(_spender, _value);
437     }
438
439     //@CTK NO_OVERFLOW
440     //@CTK NO_BUF_OVERFLOW
441     //@CTK FAIL NO_ASF
```

```
442     /*@CTK "increaseApproval correctness"
443       @tag assume_completion
444       @post !paused
445       @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
              _addedValue
446      */
447     function increaseApproval(address _spender, uint256 _addedValue) public
            whenNotPaused returns (bool){
448         return super.increaseApproval(_spender, _addedValue);
449     }
450
451     //@CTK NO_OVERFLOW
452     //@CTK NO_BUF_OVERFLOW
453     //@CTK NO_ASF
454     /*@CTK decreaseApproval0
455       @pre __return == true
456       @pre !paused
457       @pre allowed[msg.sender][_spender] <= _subtractedValue
458       @post __post.allowed[msg.sender][_spender] == 0
459      */
460     /*@CTK decreaseApproval
461       @pre __return == true
462       @pre !paused
463       @pre allowed[msg.sender][_spender] > _subtractedValue
464       @post __post.allowed[msg.sender][_spender] ==
465           allowed[msg.sender][_spender] - _subtractedValue
466      */
467     function decreaseApproval(address _spender, uint256 _subtractedValue) public
            whenNotPaused returns (bool){
468         return super.decreaseApproval(_spender, _subtractedValue);
469     }
470 }
471
472
473 /**
474  * @title TaxaNetwork token
475  *
476  * @dev PausableToken modified with coin specific setting.
477  **/
478
479 contract TaxaNetworkToken is PausableToken {
480     string public constant name = "Taxa Token";
481     string public constant symbol = "TXT";
482     uint8 public constant decimals = 18;
483
484     uint256 public constant totalSupply = 10 ** 10 * 10 ** uint256(decimals);
485
486     //@CTK NO_OVERFLOW
487     //@CTK NO_BUF_OVERFLOW
488     //@CTK NO_ASF
489     function TaxaNetworkToken() public {
490         balances[owner] = totalSupply;
491         Transfer(address(0), owner, balances[owner]);
492     }
493 }
```

File TaxaLockFoundation.sol

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

```
2
3  /**
4   * @title SafeMath
5   * @dev Math operations with safety checks that throw on error
6   */
7  library SafeMath {
8      //@CTK FAIL NO_ASF
9      /*@CTK "SafeMath mul"
10       @post ((a > 0) && (((a * b) / a) != b)) == (__reverted)
11       @post !__reverted -> __return == a * b
12       @post !__reverted == !__has_overflow
13       @post !__reverted -> !(__has_assertion_failure)
14       @post !(__has_buf_overflow)
15     */
16     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
17         if (a == 0) {
18             return 0;
19         }
20         uint256 c = a * b;
21         assert(c / a == b);
22         return c;
23     }
24
25     //@CTK FAIL NO_ASF
26     /*@CTK "SafeMath div"
27       @post b != 0 -> !__reverted
28       @post !__reverted -> __return == a / b
29       @post !__reverted -> !__has_overflow
30       @post !__reverted -> !(__has_assertion_failure)
31       @post !(__has_buf_overflow)
32     */
33     function div(uint256 a, uint256 b) internal pure returns (uint256) {
34         // assert(b > 0); // Solidity automatically throws when dividing by 0
35         uint256 c = a / b;
36         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
37         return c;
38     }
39
40     //@CTK FAIL NO_ASF
41     /*@CTK "SafeMath sub"
42       @post (a < b) == __reverted
43       @post !__reverted -> __return == a - b
44       @post !__reverted -> !__has_overflow
45       @post !__reverted -> !(__has_assertion_failure)
46       @post !(__has_buf_overflow)
47     */
48     function sub(uint256 a, uint256 b) internal pure returns (uint256) {
49         assert(b <= a);
50         return a - b;
51     }
52
53     //@CTK FAIL NO_ASF
54     /*@CTK "SafeMath add"
55       @post (a + b < a || a + b < b) == __reverted
56       @post !__reverted -> __return == a + b
57       @post !__reverted -> !__has_overflow
58       @post !(__has_buf_overflow)
59     */
```

```
60      function add(uint256 a, uint256 b) internal pure returns (uint256) {
61          uint256 c = a + b;
62          assert(c >= a);
63          return c;
64      }
65  }
66
67
68  /**
69   * @title Ownable
70   * @dev The Ownable contract has an owner address, and provides basic authorization
         control
71   * functions, this simplifies the implementation of "user permissions".
72   */
73  contract Ownable {
74      address public owner;
75
76      event OwnershipTransferred(address indexed previousOwner, address indexed newOwner
             );
77
78      /**
79       * @dev The Ownable constructor sets the original `owner` of the contract to the
             sender
80       * account.
81       */
82      constructor() public {
83          owner = msg.sender;
84      }
85
86      /**
87       * @dev Throws if called by any account other than the owner.
88       */
89      modifier onlyOwner() {
90          require(msg.sender == owner);
91          _;
92      }
93
94      /**
95       * @dev Allows the current owner to transfer control of the contract to a newOwner
             .
96       * @param newOwner The address to transfer ownership to.
97       */
98      /*@CTK transferOwnership
99        @tag assume_completion
100       @post newOwner != address(0)
101       @post __post.owner == newOwner
102      */
103     function transferOwnership(address newOwner) public onlyOwner {
104         require(newOwner != address(0));
105         emit OwnershipTransferred(owner, newOwner);
106         owner = newOwner;
107     }
108 }
109
110
111 /**
112  * @title ERC20Basic
113  * @dev Simpler version of ERC20 interface
```

```
114  * @dev see https://github.com/ethereum/EIPs/issues/179
115  */
116 contract ERC20Basic {
117   function totalSupply() public view returns (uint256);
118   function balanceOf(address who) public view returns (uint256);
119   function transfer(address to, uint256 value) public returns (bool);
120   event Transfer(address indexed from, address indexed to, uint256 value);
121 }
122
123
124 contract ERC20 is ERC20Basic {
125   function allowance(address owner, address spender)
126     public view returns (uint256);
127
128   function transferFrom(address from, address to, uint256 value)
129     public returns (bool);
130
131   function approve(address spender, uint256 value) public returns (bool);
132   event Approval(
133     address indexed owner,
134     address indexed spender,
135     uint256 value
136   );
137 }
138
139
140 /**
141  * @title SafeERC20
142  * @dev Wrappers around ERC20 operations that throw on failure.
143  * To use this library you can add a `using SafeERC20 for ERC20;` statement to your
       contract,
144  * which allows you to call the safe operations as `token.safeTransfer(...)`, etc.
145  */
146 library SafeERC20 {
147   function safeTransfer(ERC20Basic token, address to, uint256 value) internal {
148     require(token.transfer(to, value));
149   }
150
151   function safeTransferFrom(
152     ERC20 token,
153     address from,
154     address to,
155     uint256 value
156   )
157     internal
158   {
159     require(token.transferFrom(from, to, value));
160   }
161
162   function safeApprove(ERC20 token, address spender, uint256 value) internal {
163     require(token.approve(spender, value));
164   }
165 }
166
167 /**
168  * @title TokenVesting
169  * @dev A token holder contract that can release its token balance gradually like a
170  * typical vesting scheme, with a cliff and vesting period. Optionally revocable by
```

```
    the
171  * owner.
172  */
173 contract TokenVesting is Ownable {
174   using SafeMath for uint256;
175   using SafeERC20 for ERC20Basic;
176
177   event Released(uint256 amount);
178   event Revoked();
179
180   ERC20Basic public token;
181
182   // beneficiary of tokens after they are released
183   address public beneficiary;
184
185   uint256 public cliff;
186   uint256 public start;
187   uint256 public duration;
188
189   bool public revocable;
190
191   uint256 public released;
192   bool public revoked;
193   // CTK comments
194   mapping (address => uint256) public balances;
195
196   /**
197    * @dev Creates a vesting contract that vests its balance of any ERC20 token to the
198    * _beneficiary, gradually in a linear fashion until _start + _duration. By then all
199    * of the balance will have vested.
200    * @param _token address of managed token
201    * @param _beneficiary address of the beneficiary to whom vested tokens are
       transferred
202    * @param _cliff duration in seconds of the cliff in which tokens will begin to vest
203    * @param _start the time (as Unix time) at which point vesting starts
204    * @param _duration duration in seconds of the period in which the tokens will vest
205    * @param _revocable whether the vesting is revocable or not
206    */
207   /*@CTK "constructor"
208     @tag assume_completion
209     @post _beneficiary != address(0)
210     @post _cliff <= _duration
211     @post __post.beneficiary == _beneficiary
212     @post __post.revocable == _revocable
213     @post __post.duration == _duration
214     @post __post.start == _start
215     @post __post.cliff == _start + _cliff
216    */
217   constructor(
218     ERC20Basic _token,
219     address _beneficiary,
220     uint256 _start,
221     uint256 _cliff,
222     uint256 _duration,
223     bool _revocable
224   )
225     public
226   {
```

```
227     require(_beneficiary != address(0));
228     require(_cliff <= _duration);
229
230     token = _token;
231     beneficiary = _beneficiary;
232     revocable = _revocable;
233     duration = _duration;
234     cliff = _start.add(_cliff);
235     start = _start;
236   }
237
238   /**
239    * @notice Transfers vested tokens to beneficiary.
240    */
241   /*@CTK release
242     @tag assume_completion
243     @post __post.released >= released
244    */
245   function releaseToken() public {
246     uint256 unreleased = releasableAmount();
247
248     require(unreleased > 0);
249
250     released = released.add(unreleased);
251
252     token.safeTransfer(beneficiary, unreleased);
253
254     emit Released(unreleased);
255   }
256
257   /**
258    * @notice Allows the owner to revoke the vesting. Tokens already vested
259    * remain in the contract, the rest are returned to the owner.
260    */
261   /*@CTK revoke
262     @tag assume_completion
263     @post owner == msg.sender
264     @post revocable == true
265     @post revoked == false
266     @post __post.revoked == true
267     */
268   function revokeToken() public onlyOwner {
269     require(revocable);
270     require(!revoked);
271
272     uint256 balance = balances[token];
273
274     uint256 unreleased = releasableAmount();
275     uint256 refund = balance.sub(unreleased);
276
277     revoked = true;
278
279     token.safeTransfer(owner, refund);
280
281     emit Revoked();
282   }
283
284   /**
```

```
285     * @dev Calculates the amount that has already vested but hasn't been released yet.
286     */
287   function releasableAmount() public view returns (uint256) {
288     return vestedAmount().sub(released);
289   }
290
291   /**
292    * @dev Calculates the amount that has already vested.
293    */
294   /*@CTK vestedAmount_not_cliff_yet
295     @tag assume_completion
296     @pre now < cliff
297     @post __return == 0
298    */
299   /*@CTK remainingAmountAreRevokedOrReleased
300     @tag assume_completion
301     @pre (now >= start + duration || revoked) && (now >= cliff)
302     @post __return == balances[token] + released
303    */
304   /*CTK partialAmountReleased
305     @tag assume_completion
306     @pre now >= cliff && now < start + duration && !revoked
307     @post __return == (balances[token] + released) * (now - start) / duration
308    */
309   function vestedAmount() public view returns (uint256) {
310     uint256 currentBalance = balances[token];
311     uint256 totalBalance = currentBalance.add(released);
312
313     if (block.timestamp < cliff) {
314       return 0;
315     } else if (block.timestamp >= start.add(duration) || revoked) {
316       return totalBalance;
317     } else {
318       return totalBalance.mul(block.timestamp.sub(start)).div(duration);
319     }
320   }
321 }
```