CertiK Audit Report For WINk

Request Date: 2019-07-23
Revision Date: 2019-08-06
Platform Name: TRON

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

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

Vulnerability Classification

For every issue found, CertiK categorizes them into 3 buckets based on its risk level:

Critical

The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.

Medium

The code implementation does not match the specification at certain conditions, or it could affect the security standard by lost of access control.

Low

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

Testing Summary

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.	1	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

"tx.origin" for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **MinterRole.sol**
  `065d0547ceca01b34aa6735d151925ef2c917a58061e61083ccf6ecaad3d0e31`
- **Roles.sol**
  `eaa4e96f6ce1835ad5fe053780c6d5fc809b55380065395d62f286a2820475b4`
- **SafeMath.sol**
  `9e2ab1725d9b0ac78b859ac580fc9d1907a5145b28bf975d49812b0fd3b345e5`
- **ITRC20.sol**
  `20702cea05216f38212e588bdfd19001e9f7f7ac6e1fa8f46a103d5e40c88126`
- **TRC20.sol**
  `aec1314beb5ea6dc46714ea9c7ffd5604387313b11ebd7fe23555246ef98d375`
- **TRC20Burnable.sol**
  `a8160adf350f75ea2eeb55587af2e680e188faeccd853e025dff47be8e748278`
- **TRC20Capped.sol**
  `cd4e6fce0c14bde830eb22b827f3c6374f73b3fc51eb0f482cab56bf9c5ddd56`
- **TRC20Detailed.sol**
  `49a767e4601647500af2436342d3dd99e471ad38348bce5932e3a68c2018302a`
- **TRC20Mintable.sol**
  `a9b5fc7887d42329b4e5749ef48cd1c19b152e6884166705f0e7d8e7291fe07e`
- **WINK.sol**
  `1bc364e229302eeb5ce25ceefffe88b42e2ad641b3479f4f01940868c80bad34`

Summary

CertiK was chosen by TRON to audit the design and implementation of its WINk smart contract. 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.

Recommendations

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

TRC20.sol

- INFO Ideally TRC20 should be minimally defined and implemented as TRC10 described. Currently it contains extra logic such as `_mint()`, `_burn()` and `_burnFrom()`, consider to put those features accordingly to `TRC20Burnable` and `TRC20Mintable`.
- INFO `transfer()`: Recommend adding `require(_balances[from] >= value, ...)` check.
- INFO `transferFrom()`: Recommend adding `require(_allowed[from][msg.sender] >= value, ...)` check.

TRC20Capped.sol TRC20Detailed.sol TRC20Mintable.sol TRC20Burnable.sol

- INFO These files are included in repo however not used.

WINK.sol

- INFO Consider to inherit from `TRC20Burnable`, `TRC20Detailed` and `TRC20Mintable` to make it more componentized design pattern. E.g.,

```
contract WINK is TRC20Mintable, TRC20Burnable, TRC20Detailed {
    uint private _initialSupply;
    constructor(string _name, string _symbol, uint8 _decimals, uint
        _initialSupply)
    TRC20Detailed(_name, _symbol, _decimals)
    TRC20Burnable()
    TRC20Mintable() public {
        super._mint(msg.sender, _initialSupply);
    }
}
```

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

```
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

INSECURE_COMPILER_VERSION

Line 1 in File WINK.sol

```
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

INSECURE_COMPILER_VERSION

Line 1 in File TRC20.sol

```
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

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

SafeMath mul zero

29, Jul 2019
19.42 ms

Line 12-17 in File SafeMath.sol

```
12  /*@CTK "SafeMath mul zero"
13    @tag spec
14    @tag is_pure
15    @pre (a == 0)
16    @post __return == 0
17  */
```

Line 28-40 in File SafeMath.sol

```
28  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
29      // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
30      // benefit is lost if 'b' is also tested.
31      // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
32      if (a == 0) {
33          return 0;
34      }
35
36      uint256 c = a * b;
37      require(c / a == b);
38
39      return c;
40  }
```

The code meets the specification.

Formal Verification Request 2

SafeMath mul nonzero

29, Jul 2019
320.28 ms

Line 18-27 in File SafeMath.sol

```
18  /*@CTK "SafeMath mul nonzero"
19    @tag spec
20    @tag is_pure
21    @pre (a != 0)
22    @post (a * b / a != b) == __reverted
23    @post !__reverted -> __return == a * b
24    @post !__reverted -> !__has_overflow
25    @post !__reverted -> !__has_assertion_failure
26    @post !(__has_buf_overflow)
27  */
```

Line 28-40 in File SafeMath.sol

```
28  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
29      // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
30      // benefit is lost if 'b' is also tested.
```

```
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }
```

The code meets the specification.

Formal Verification Request 3

SafeMath div

29, Jul 2019
12.64 ms

Line 45-53 in File SafeMath.sol

```
45  /*@CTK "SafeMath div"
46    @tag spec
47    @tag is_pure
48    @post (b == 0) == __reverted
49    @post !__reverted -> __return == a / b
50    @post !__reverted -> !__has_overflow
51    @post !__reverted -> !__has_assertion_failure
52    @post !(__has_buf_overflow)
53  */
```

Line 54-60 in File SafeMath.sol

```
54  function div(uint256 a, uint256 b) internal pure returns (uint256) {
55      require(b > 0); // Solidity only automatically asserts when dividing by 0
56      uint256 c = a / b;
57      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
58
59      return c;
60  }
```

The code meets the specification.

Formal Verification Request 4

SafeMath sub

29, Jul 2019
11.65 ms

Line 65-73 in File SafeMath.sol

```
65  /*@CTK "SafeMath sub"
66    @tag spec
67    @tag is_pure
```

```
68      @post (b > a) == __reverted
69      @post !__reverted -> __return == a - b
70      @post !__reverted -> !__has_overflow
71      @post !__reverted -> !__has_assertion_failure
72      @post !(__has_buf_overflow)
73    */
```

Line 74-79 in File SafeMath.sol

```
74    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
75        require(b <= a);
76        uint256 c = a - b;
77
78        return c;
79    }
```

The code meets the specification.

Formal Verification Request 5

SafeMath add

29, Jul 2019
15.59 ms

Line 84-92 in File SafeMath.sol

```
84    /*@CTK "SafeMath add"
85      @tag spec
86      @tag is_pure
87      @post (a + b < a || a + b < b) == __reverted
88      @post !__reverted -> __return == a + b
89      @post !__reverted -> !__has_overflow
90      @post !__reverted -> !__has_assertion_failure
91      @post !(__has_buf_overflow)
92    */
```

Line 93-98 in File SafeMath.sol

```
93    function add(uint256 a, uint256 b) internal pure returns (uint256) {
94        uint256 c = a + b;
95        require(c >= a);
96
97        return c;
98    }
```

The code meets the specification.

Formal Verification Request 6

SafeMath mod

29, Jul 2019
12.06 ms

Line 104-112 in File SafeMath.sol

```
104 /*@CTK "SafeMath mod"
105   @tag spec
106   @tag is_pure
107   @post (b == 0) == __reverted
108   @post !__reverted -> __return == a % b
109   @post !__reverted -> !__has_overflow
110   @post !__reverted -> !__has_assertion_failure
111   @post !(__has_buf_overflow)
112 */
```

Line 113-116 in File SafeMath.sol

```
113 function mod(uint256 a, uint256 b) internal pure returns (uint256) {
114     require(b != 0);
115     return a % b;
116 }
```

✅ The code meets the specification.

Formal Verification Request 7

If method completes, integer overflow would not happen.

📅 29, Jul 2019
⏱ 10.24 ms

Line 21 in File WINK.sol

```
21 //@CTK NO_OVERFLOW
```

Line 30-34 in File WINK.sol

```
30 constructor (string name, string symbol, uint8 decimals) public {
31     _name = name;
32     _symbol = symbol;
33     _decimals = decimals;
34 }
```

✅ The code meets the specification.

Formal Verification Request 8

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

📅 29, Jul 2019
⏱ 0.37 ms

Line 22 in File WINK.sol

```
22 //@CTK NO_BUF_OVERFLOW
```

Line 30-34 in File WINK.sol

```
30 constructor (string name, string symbol, uint8 decimals) public {
31     _name = name;
32     _symbol = symbol;
33     _decimals = decimals;
34 }
```

The code meets the specification.

Formal Verification Request 9

Method will not encounter an assertion failure.

29, Jul 2019
0.3 ms

Line 23 in File WINK.sol

```
23    //@CTK NO_ASF
```

Line 30-34 in File WINK.sol

```
30    constructor (string name, string symbol, uint8 decimals) public {
31        _name = name;
32        _symbol = symbol;
33        _decimals = decimals;
34    }
```

The code meets the specification.

Formal Verification Request 10

WINK

29, Jul 2019
0.95 ms

Line 24-29 in File WINK.sol

```
24    /*@CTK WINK
25      @tag assume_completion
26      @post __post._name == name
27      @post __post._symbol == symbol
28      @post __post._decimals == decimals
29    */
```

Line 30-34 in File WINK.sol

```
30    constructor (string name, string symbol, uint8 decimals) public {
31        _name = name;
32        _symbol = symbol;
33        _decimals = decimals;
34    }
```

The code meets the specification.

Formal Verification Request 11

If method completes, integer overflow would not happen.

29, Jul 2019
4.56 ms

Line 39 in File WINK.sol

```
39    //@CTK NO_OVERFLOW
```

Line 46-48 in File WINK.sol

```
46    function name() public view returns (string) {
47        return _name;
48    }
```

The code meets the specification.

Formal Verification Request 12

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

29, Jul 2019
0.31 ms

Line 40 in File WINK.sol

```
40    //@CTK NO_BUF_OVERFLOW
```

Line 46-48 in File WINK.sol

```
46    function name() public view returns (string) {
47        return _name;
48    }
```

The code meets the specification.

Formal Verification Request 13

Method will not encounter an assertion failure.

29, Jul 2019
0.29 ms

Line 41 in File WINK.sol

```
41    //@CTK NO_ASF
```

Line 46-48 in File WINK.sol

```
46    function name() public view returns (string) {
47        return _name;
48    }
```

The code meets the specification.

Formal Verification Request 14

name

29, Jul 2019
0.29 ms

Line 42-45 in File WINK.sol

```
42  /*@CTK name
43    @tag assume_completion
44    @post __return == _name
45  */
```

Line 46-48 in File WINK.sol

```
46  function name() public view returns (string) {
47      return _name;
48  }
```

The code meets the specification.

Formal Verification Request 15

If method completes, integer overflow would not happen.

29, Jul 2019
4.78 ms

Line 53 in File WINK.sol

```
53  //@CTK NO_OVERFLOW
```

Line 60-62 in File WINK.sol

```
60  function symbol() public view returns (string) {
61      return _symbol;
62  }
```

The code meets the specification.

Formal Verification Request 16

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

29, Jul 2019
0.31 ms

Line 54 in File WINK.sol

```
54  //@CTK NO_BUF_OVERFLOW
```

Line 60-62 in File WINK.sol

```
60  function symbol() public view returns (string) {
61      return _symbol;
62  }
```

The code meets the specification.

Formal Verification Request 17

Method will not encounter an assertion failure.

29, Jul 2019
0.3 ms

Line 55 in File WINK.sol

```
55    //@CTK NO_ASF
```

Line 60-62 in File WINK.sol

```
60    function symbol() public view returns (string) {
61        return _symbol;
62    }
```

The code meets the specification.

Formal Verification Request 18

symbol

29, Jul 2019
0.3 ms

Line 56-59 in File WINK.sol

```
56    /*@CTK symbol
57      @tag assume_completion
58      @post __return == _symbol
59    */
```

Line 60-62 in File WINK.sol

```
60    function symbol() public view returns (string) {
61        return _symbol;
62    }
```

The code meets the specification.

Formal Verification Request 19

If method completes, integer overflow would not happen.

29, Jul 2019
5.86 ms

Line 67 in File WINK.sol

```
67    //@CTK NO_OVERFLOW
```

Line 74-76 in File WINK.sol

```
74    function decimals() public view returns (uint8) {
75        return _decimals;
76    }
```

The code meets the specification.

Formal Verification Request 20

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

29, Jul 2019
0.39 ms

Line 68 in File WINK.sol

```
68    //@CTK NO_BUF_OVERFLOW
```

Line 74-76 in File WINK.sol

```
74    function decimals() public view returns (uint8) {
75        return _decimals;
76    }
```

The code meets the specification.

Formal Verification Request 21

Method will not encounter an assertion failure.

29, Jul 2019
0.34 ms

Line 69 in File WINK.sol

```
69    //@CTK NO_ASF
```

Line 74-76 in File WINK.sol

```
74    function decimals() public view returns (uint8) {
75        return _decimals;
76    }
```

The code meets the specification.

Formal Verification Request 22

decimals

29, Jul 2019
0.35 ms

Line 70-73 in File WINK.sol

```
70    /*@CTK decimals
71      @tag assume_completion
72      @post __return == _decimals
73    */
```

Line 74-76 in File WINK.sol

```
74    function decimals() public view returns (uint8) {
75        return _decimals;
76    }
```

The code meets the specification.

Formal Verification Request 23

If method completes, integer overflow would not happen.

29, Jul 2019
5.75 ms

Line 25 in File TRC20.sol

```
25    //@CTK NO_OVERFLOW
```

Line 32-34 in File TRC20.sol

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

The code meets the specification.

Formal Verification Request 24

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

29, Jul 2019
0.36 ms

Line 26 in File TRC20.sol

```
26    //@CTK NO_BUF_OVERFLOW
```

Line 32-34 in File TRC20.sol

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

The code meets the specification.

Formal Verification Request 25

Method will not encounter an assertion failure.

29, Jul 2019
0.36 ms

Line 27 in File TRC20.sol

```
27    //@CTK NO_ASF
```

Line 32-34 in File TRC20.sol

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

The code meets the specification.

Formal Verification Request 26

totalSupply

29, Jul 2019
0.36 ms

Line 28-31 in File TRC20.sol

```
28  /*@CTK totalSupply
29      @tag assume_completion
30      @post (__return) == (_totalSupply)
31  */
```

Line 32-34 in File TRC20.sol

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

The code meets the specification.

Formal Verification Request 27

If method completes, integer overflow would not happen.

29, Jul 2019
5.54 ms

Line 41 in File TRC20.sol

```
41  //@CTK NO_OVERFLOW
```

Line 47-49 in File TRC20.sol

```
47  function balanceOf(address owner) public view returns (uint256) {
48      return _balances[owner];
49  }
```

The code meets the specification.

Formal Verification Request 28

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

29, Jul 2019
0.33 ms

Line 42 in File TRC20.sol

```
42  //@CTK NO_BUF_OVERFLOW
```

Line 47-49 in File TRC20.sol

```
47  function balanceOf(address owner) public view returns (uint256) {
48      return _balances[owner];
49  }
```

The code meets the specification.

Formal Verification Request 29

Method will not encounter an assertion failure.

29, Jul 2019
0.32 ms

Line 43 in File TRC20.sol

```
43    //@CTK NO_ASF
```

Line 47-49 in File TRC20.sol

```
47    function balanceOf(address owner) public view returns (uint256) {
48        return _balances[owner];
49    }
```

The code meets the specification.

Formal Verification Request 30

balanceOf

29, Jul 2019
0.34 ms

Line 44-46 in File TRC20.sol

```
44    /*@CTK balanceOf
45      @post __return == __post._balances[owner]
46     */
```

Line 47-49 in File TRC20.sol

```
47    function balanceOf(address owner) public view returns (uint256) {
48        return _balances[owner];
49    }
```

The code meets the specification.

Formal Verification Request 31

If method completes, integer overflow would not happen.

29, Jul 2019
5.49 ms

Line 57 in File TRC20.sol

```
57    //@CTK NO_OVERFLOW
```

Line 63-72 in File TRC20.sol

```
63    function allowance(
64        address owner,
65        address spender
66    )
67    public
```

```
    view
    returns (uint256)
    {
        return _allowed[owner][spender];
    }
```

The code meets the specification.

Formal Verification Request 32

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

29, Jul 2019
0.33 ms

Line 58 in File TRC20.sol

```
58  //@CTK NO_BUF_OVERFLOW
```

Line 63-72 in File TRC20.sol

```
63  function allowance(
64      address owner,
65      address spender
66  )
67  public
68  view
69  returns (uint256)
70  {
71      return _allowed[owner][spender];
72  }
```

The code meets the specification.

Formal Verification Request 33

Method will not encounter an assertion failure.

29, Jul 2019
0.34 ms

Line 59 in File TRC20.sol

```
59  //@CTK NO_ASF
```

Line 63-72 in File TRC20.sol

```
63  function allowance(
64      address owner,
65      address spender
66  )
67  public
68  view
69  returns (uint256)
70  {
71      return _allowed[owner][spender];
72  }
```

The code meets the specification.

Formal Verification Request 34

allowance correctness

29, Jul 2019
0.34 ms

Line 60-62 in File TRC20.sol

```
60  /*@CTK "allowance correctness"
61    @post __return == _allowed[owner][spender]
62   */
```

Line 63-72 in File TRC20.sol

```
63  function allowance(
64      address owner,
65      address spender
66  )
67  public
68  view
69  returns (uint256)
70  {
71      return _allowed[owner][spender];
72  }
```

The code meets the specification.

Formal Verification Request 35

If method completes, integer overflow would not happen.

29, Jul 2019
77.94 ms

Line 79 in File TRC20.sol

```
79  //@CTK NO_OVERFLOW
```

Line 89-92 in File TRC20.sol

```
89  function transfer(address to, uint256 value) public returns (bool) {
90      _transfer(msg.sender, to, value);
91      return true;
92  }
```

The code meets the specification.

Formal Verification Request 36

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

29, Jul 2019
5.8 ms

Line 80 in File TRC20.sol

```
80    //@CTK NO_BUF_OVERFLOW
```

Line 89-92 in File TRC20.sol

```
89    function transfer(address to, uint256 value) public returns (bool) {
90        _transfer(msg.sender, to, value);
91        return true;
92    }
```

The code meets the specification.

Formal Verification Request 37

transfer

29, Jul 2019
28.82 ms

Line 81-88 in File TRC20.sol

```
81    /*@CTK transfer
82      @tag assume_completion
83      @pre msg.sender != to
84      @post to != address(0)
85      @post value <= _balances[msg.sender]
86      @post __post._balances[msg.sender] == _balances[msg.sender] - value
87      @post __post._balances[to] == _balances[to] + value
88     */
```

Line 89-92 in File TRC20.sol

```
89    function transfer(address to, uint256 value) public returns (bool) {
90        _transfer(msg.sender, to, value);
91        return true;
92    }
```

The code meets the specification.

Formal Verification Request 38

If method completes, integer overflow would not happen.

29, Jul 2019
15.68 ms

Line 103 in File TRC20.sol

```
103   //@CTK NO_OVERFLOW
```

Line 117-123 in File TRC20.sol

```
117   function approve(address spender, uint256 value) public returns (bool) {
118       require(spender != address(0));
119
120       _allowed[msg.sender][spender] = value;
```

```
121         emit Approval(msg.sender, spender, value);
122         return true;
123     }
```

The code meets the specification.

Formal Verification Request 39

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

29, Jul 2019
0.48 ms

Line 104 in File TRC20.sol

```
104     //@CTK NO_BUF_OVERFLOW
```

Line 117-123 in File TRC20.sol

```
117     function approve(address spender, uint256 value) public returns (bool) {
118         require(spender != address(0));
119 
120         _allowed[msg.sender][spender] = value;
121         emit Approval(msg.sender, spender, value);
122         return true;
123     }
```

The code meets the specification.

Formal Verification Request 40

Method will not encounter an assertion failure.

29, Jul 2019
0.44 ms

Line 105 in File TRC20.sol

```
105     //@CTK NO_ASF
```

Line 117-123 in File TRC20.sol

```
117     function approve(address spender, uint256 value) public returns (bool) {
118         require(spender != address(0));
119 
120         _allowed[msg.sender][spender] = value;
121         emit Approval(msg.sender, spender, value);
122         return true;
123     }
```

The code meets the specification.

Formal Verification Request 41

approve normal spender

29, Jul 2019
2.01 ms

Line 106-109 in File TRC20.sol

```
106 /*@CTK "approve normal spender"
107   @pre spender != address(0)
108   @post __post._allowed[msg.sender][spender] == value
109  */
```

Line 117-123 in File TRC20.sol

```
117 function approve(address spender, uint256 value) public returns (bool) {
118     require(spender != address(0));
119
120     _allowed[msg.sender][spender] = value;
121     emit Approval(msg.sender, spender, value);
122     return true;
123 }
```

The code meets the specification.

Formal Verification Request 42

If method completes, integer overflow would not happen.

29, Jul 2019
15.68 ms

Line 110 in File TRC20.sol

```
110 //@CTK NO_OVERFLOW
```

Line 117-123 in File TRC20.sol

```
117 function approve(address spender, uint256 value) public returns (bool) {
118     require(spender != address(0));
119
120     _allowed[msg.sender][spender] = value;
121     emit Approval(msg.sender, spender, value);
122     return true;
123 }
```

The code meets the specification.

Formal Verification Request 43

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

29, Jul 2019
0.48 ms

Line 111 in File TRC20.sol

```
111     //@CTK NO_BUF_OVERFLOW
```

Line 117-123 in File TRC20.sol

```
117     function approve(address spender, uint256 value) public returns (bool) {
118         require(spender != address(0));
119
120         _allowed[msg.sender][spender] = value;
121         emit Approval(msg.sender, spender, value);
122         return true;
123     }
```

The code meets the specification.

Formal Verification Request 44

Method will not encounter an assertion failure.

29, Jul 2019
0.44 ms

Line 112 in File TRC20.sol

```
112     //@CTK NO_ASF
```

Line 117-123 in File TRC20.sol

```
117     function approve(address spender, uint256 value) public returns (bool) {
118         require(spender != address(0));
119
120         _allowed[msg.sender][spender] = value;
121         emit Approval(msg.sender, spender, value);
122         return true;
123     }
```

The code meets the specification.

Formal Verification Request 45

approve zero spender

29, Jul 2019
0.83 ms

Line 113-116 in File TRC20.sol

```
113     /*@CTK "approve zero spender"
114       @pre spender == address(0)
115       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender]
116      */
```

Line 117-123 in File TRC20.sol

```
117     function approve(address spender, uint256 value) public returns (bool) {
118         require(spender != address(0));
119
120         _allowed[msg.sender][spender] = value;
121         emit Approval(msg.sender, spender, value);
```

```
        return true;
    }
```

The code meets the specification.

Formal Verification Request 46

If method completes, integer overflow would not happen.

29, Jul 2019
60.72 ms

Line 131 in File TRC20.sol

```
    //@CTK NO_OVERFLOW
```

Line 142-153 in File TRC20.sol

```
    function transferFrom(
        address from,
        address to,
        uint256 value
    )
    public
    returns (bool)
    {
        _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
        _transfer(from, to, value);
        return true;
    }
```

The code meets the specification.

Formal Verification Request 47

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

29, Jul 2019
6.35 ms

Line 132 in File TRC20.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 142-153 in File TRC20.sol

```
    function transferFrom(
        address from,
        address to,
        uint256 value
    )
    public
    returns (bool)
    {
        _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
        _transfer(from, to, value);
        return true;
    }
```

The code meets the specification.

Formal Verification Request 48

transferFrom correctness

29, Jul 2019
139.94 ms

Line 133-141 in File TRC20.sol

```
133 /*@CTK "transferFrom correctness"
134   @tag assume_completion
135   @post to != 0x0
136   @post value <= _balances[from] && value <= _allowed[from][msg.sender]
137   @post to != from -> __post._balances[from] == _balances[from] - value
138   @post to != from -> __post._balances[to] == _balances[to] + value
139   @post to == from -> __post._balances[from] == _balances[from]
140   @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
141  */
```

Line 142-153 in File TRC20.sol

```
142 function transferFrom(
143     address from,
144     address to,
145     uint256 value
146 )
147 public
148 returns (bool)
149 {
150     _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
151     _transfer(from, to, value);
152     return true;
153 }
```

The code meets the specification.

Formal Verification Request 49

If method completes, integer overflow would not happen.

29, Jul 2019
19.67 ms

Line 164 in File TRC20.sol

```
164 //@CTK NO_OVERFLOW
```

Line 171-184 in File TRC20.sol

```
171 function increaseAllowance(
172     address spender,
173     uint256 addedValue
174 )
175 public
176 returns (bool)
```

```
177     {
178         require(spender != address(0));
179
180         _allowed[msg.sender][spender] = (
181         _allowed[msg.sender][spender].add(addedValue));
182         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
183         return true;
184     }
```

The code meets the specification.

Formal Verification Request 50

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

29, Jul 2019
0.45 ms

Line 165 in File TRC20.sol

```
165     //@CTK NO_BUF_OVERFLOW
```

Line 171-184 in File TRC20.sol

```
171     function increaseAllowance(
172         address spender,
173         uint256 addedValue
174     )
175     public
176     returns (bool)
177     {
178         require(spender != address(0));
179
180         _allowed[msg.sender][spender] = (
181         _allowed[msg.sender][spender].add(addedValue));
182         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
183         return true;
184     }
```

The code meets the specification.

Formal Verification Request 51

increaseAllowance correctness

29, Jul 2019
2.89 ms

Line 166-170 in File TRC20.sol

```
166     /*@CTK "increaseAllowance correctness"
167       @tag assume_completion
168       @post spender != address(0)
169       @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
              addedValue
170      */
```

Line 171-184 in File TRC20.sol

```
171 function increaseAllowance(
172     address spender,
173     uint256 addedValue
174 )
175 public
176 returns (bool)
177 {
178     require(spender != address(0));
179
180     _allowed[msg.sender][spender] = (
181     _allowed[msg.sender][spender].add(addedValue));
182     emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
183     return true;
184 }
```

The code meets the specification.

Formal Verification Request 52

If method completes, integer overflow would not happen.

29, Jul 2019
20.51 ms

Line 195 in File TRC20.sol

```
195 //@CTK NO_OVERFLOW
```

Line 203-216 in File TRC20.sol

```
203 function decreaseAllowance(
204     address spender,
205     uint256 subtractedValue
206 )
207 public
208 returns (bool)
209 {
210     require(spender != address(0));
211
212     _allowed[msg.sender][spender] = (
213     _allowed[msg.sender][spender].sub(subtractedValue));
214     emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
215     return true;
216 }
```

The code meets the specification.

Formal Verification Request 53

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

29, Jul 2019
0.56 ms

Line 196 in File TRC20.sol

```
196     //@CTK NO_BUF_OVERFLOW
```

Line 203-216 in File TRC20.sol

```
203     function decreaseAllowance(
204         address spender,
205         uint256 subtractedValue
206     )
207     public
208     returns (bool)
209     {
210         require(spender != address(0));
211
212         _allowed[msg.sender][spender] = (
213         _allowed[msg.sender][spender].sub(subtractedValue));
214         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
215         return true;
216     }
```

The code meets the specification.

Formal Verification Request 54

decreaseAllowance

29, Jul 2019
2.08 ms

Line 197-202 in File TRC20.sol

```
197     /*@CTK decreaseAllowance
198       @pre _allowed[msg.sender][spender] >= subtractedValue
199       @pre spender != address(0)
200       @post __post._allowed[msg.sender][spender] ==
201           _allowed[msg.sender][spender] - subtractedValue
202      */
```

Line 203-216 in File TRC20.sol

```
203     function decreaseAllowance(
204         address spender,
205         uint256 subtractedValue
206     )
207     public
208     returns (bool)
209     {
210         require(spender != address(0));
211
212         _allowed[msg.sender][spender] = (
213         _allowed[msg.sender][spender].sub(subtractedValue));
214         emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
215         return true;
216     }
```

The code meets the specification.

Formal Verification Request 55

If method completes, integer overflow would not happen.

29, Jul 2019
5.13 ms

Line 224 in File TRC20.sol

```
224    //@CTK NO_OVERFLOW
```

Line 243-249 in File TRC20.sol

```
243    function _transfer(address from, address to, uint256 value) internal {
244        require(to != address(0));
245
246        _balances[from] = _balances[from].sub(value);
247        _balances[to] = _balances[to].add(value);
248        emit Transfer(from, to, value);
249    }
```

The code meets the specification.

Formal Verification Request 56

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

29, Jul 2019
5.84 ms

Line 225 in File TRC20.sol

```
225    //@CTK NO_BUF_OVERFLOW
```

Line 243-249 in File TRC20.sol

```
243    function _transfer(address from, address to, uint256 value) internal {
244        require(to != address(0));
245
246        _balances[from] = _balances[from].sub(value);
247        _balances[to] = _balances[to].add(value);
248        emit Transfer(from, to, value);
249    }
```

The code meets the specification.

Formal Verification Request 57

_transfer_other

29, Jul 2019
31.99 ms

Line 226-233 in File TRC20.sol

```
226     /*@CTK _transfer_other
227       @tag assume_completion
228       @pre from != to
229       @pre value <= _balances[from]
230       @post to != address(0)
231       @post __post._balances[from] == _balances[from] - value
232       @post __post._balances[to] == _balances[to] + value
233      */
```

Line 243-249 in File TRC20.sol

```
243     function _transfer(address from, address to, uint256 value) internal {
244         require(to != address(0));
245
246         _balances[from] = _balances[from].sub(value);
247         _balances[to] = _balances[to].add(value);
248         emit Transfer(from, to, value);
249     }
```

The code meets the specification.

Formal Verification Request 58

If method completes, integer overflow would not happen.

29, Jul 2019
5.13 ms

Line 234 in File TRC20.sol

```
234     //@CTK NO_OVERFLOW
```

Line 243-249 in File TRC20.sol

```
243     function _transfer(address from, address to, uint256 value) internal {
244         require(to != address(0));
245
246         _balances[from] = _balances[from].sub(value);
247         _balances[to] = _balances[to].add(value);
248         emit Transfer(from, to, value);
249     }
```

The code meets the specification.

Formal Verification Request 59

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

29, Jul 2019
5.84 ms

Line 235 in File TRC20.sol

```
235     //@CTK NO_BUF_OVERFLOW
```

Line 243-249 in File TRC20.sol

```
243     function _transfer(address from, address to, uint256 value) internal {
244         require(to != address(0));
245
246         _balances[from] = _balances[from].sub(value);
247         _balances[to] = _balances[to].add(value);
248         emit Transfer(from, to, value);
249     }
```

The code meets the specification.

Formal Verification Request 60

_transfer_self

29, Jul 2019
19.62 ms

Line 236-242 in File TRC20.sol

```
236     /*@CTK _transfer_self
237       @tag assume_completion
238       @pre from == to
239       @post to != address(0)
240       @post __post._balances[from] == _balances[from]
241       @post __post._balances[to] == _balances[to]
242      */
```

Line 243-249 in File TRC20.sol

```
243     function _transfer(address from, address to, uint256 value) internal {
244         require(to != address(0));
245
246         _balances[from] = _balances[from].sub(value);
247         _balances[to] = _balances[to].add(value);
248         emit Transfer(from, to, value);
249     }
```

The code meets the specification.

Formal Verification Request 61

If method completes, integer overflow would not happen.

29, Jul 2019
34.71 ms

Line 258 in File TRC20.sol

```
258     //@CTK NO_OVERFLOW
```

Line 266-272 in File TRC20.sol

```
266     function _mint(address account, uint256 value) internal {
267         require(account != address(0));
268
269         _totalSupply = _totalSupply.add(value);
270         _balances[account] = _balances[account].add(value);
```

```
        emit Transfer(address(0), account, value);
    }
```

The code meets the specification.

Formal Verification Request 62

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

29, Jul 2019
5.13 ms

Line 259 in File TRC20.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 266-272 in File TRC20.sol

```
    function _mint(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.add(value);
        _balances[account] = _balances[account].add(value);
        emit Transfer(address(0), account, value);
    }
```

The code meets the specification.

Formal Verification Request 63

mint

29, Jul 2019
30.5 ms

Line 260-265 in File TRC20.sol

```
    /*@CTK mint
      @tag assume_completion
      @post (account != address(0))
      @post (__post._totalSupply) == ((_totalSupply) + (value))
      @post (__post._balances[account]) == ((_balances[account]) + (value))
    */
```

Line 266-272 in File TRC20.sol

```
    function _mint(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.add(value);
        _balances[account] = _balances[account].add(value);
        emit Transfer(address(0), account, value);
    }
```

The code meets the specification.

Formal Verification Request 64

If method completes, integer overflow would not happen.

29, Jul 2019
29.97 ms

Line 281 in File TRC20.sol

```
281    //@CTK NO_OVERFLOW
```

Line 289-295 in File TRC20.sol

```
289    function _burn(address account, uint256 value) internal {
290        require(account != address(0));
291
292        _totalSupply = _totalSupply.sub(value);
293        _balances[account] = _balances[account].sub(value);
294        emit Transfer(account, address(0), value);
295    }
```

The code meets the specification.

Formal Verification Request 65

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

29, Jul 2019
4.57 ms

Line 282 in File TRC20.sol

```
282    //@CTK NO_BUF_OVERFLOW
```

Line 289-295 in File TRC20.sol

```
289    function _burn(address account, uint256 value) internal {
290        require(account != address(0));
291
292        _totalSupply = _totalSupply.sub(value);
293        _balances[account] = _balances[account].sub(value);
294        emit Transfer(account, address(0), value);
295    }
```

The code meets the specification.

Formal Verification Request 66

_burn

29, Jul 2019
34.08 ms

Line 283-288 in File TRC20.sol

```
283 /*@CTK _burn
284   @tag assume_completion
285   @post (value <= _balances[account])
286   @post (__post._totalSupply) == (_totalSupply - value)
287   @post (__post._balances[account]) == (_balances[account] - value)
288 */
```

Line 289-295 in File TRC20.sol

```
289 function _burn(address account, uint256 value) internal {
290     require(account != address(0));
291
292     _totalSupply = _totalSupply.sub(value);
293     _balances[account] = _balances[account].sub(value);
294     emit Transfer(account, address(0), value);
295 }
```

The code meets the specification.

Formal Verification Request 67

If method completes, integer overflow would not happen.

29, Jul 2019
54.47 ms

Line 304 in File TRC20.sol

```
304 //@CTK NO_OVERFLOW
```

Line 314-320 in File TRC20.sol

```
314 function _burnFrom(address account, uint256 value) internal {
315     // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be
            accepted,
316     // this function needs to emit an event with the updated approval.
317     _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
318         value);
319     _burn(account, value);
320 }
```

The code meets the specification.

Formal Verification Request 68

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

29, Jul 2019
7.8 ms

Line 305 in File TRC20.sol

```
305 //@CTK NO_BUF_OVERFLOW
```

Line 314-320 in File TRC20.sol

```
    function _burnFrom(address account, uint256 value) internal {
        // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted,
        // this function needs to emit an event with the updated approval.
        _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(value);
        _burn(account, value);
    }
```

The code meets the specification.

Formal Verification Request 69

_burnFrom

29, Jul 2019
127.92 ms

Line 306-313 in File TRC20.sol

```
    /*@CTK _burnFrom
      @tag assume_completion
      @post (value <= _allowed[account][msg.sender])
      @post (value <= _balances[account])
      @post (__post._allowed[account][msg.sender]) == (_allowed[account][msg.sender] - value)
      @post (__post._balances[account]) == (_balances[account] - value)
      @post __post._totalSupply == (_totalSupply - value)
    */
```

Line 314-320 in File TRC20.sol

```
    function _burnFrom(address account, uint256 value) internal {
        // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted,
        // this function needs to emit an event with the updated approval.
        _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(value);
        _burn(account, value);
    }
```

The code meets the specification.

Source Code with CertiK Labels

File utils/SafeMath.sol

```
pragma solidity ^0.4.23;

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

    /**
     * @dev Multiplies two numbers, reverts on overflow.
     */
    /*@CTK "SafeMath mul zero"
      @tag spec
      @tag is_pure
      @pre (a == 0)
      @post __return == 0
    */
    /*@CTK "SafeMath mul nonzero"
      @tag spec
      @tag is_pure
      @pre (a != 0)
      @post (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) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }

    /**
     * @dev Integer division of two numbers truncating the quotient, reverts on
       division by zero.
     */
    /*@CTK "SafeMath div"
      @tag spec
      @tag is_pure
      @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) {
        require(b > 0); // Solidity only automatically asserts 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;
    }

    /**
     * @dev Subtracts two numbers, reverts on overflow (i.e. if subtrahend is greater
         than minuend).
     */
    /*@CTK "SafeMath sub"
      @tag spec
      @tag is_pure
      @post (b > a) == __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) {
        require(b <= a);
        uint256 c = a - b;

        return c;
    }

    /**
     * @dev Adds two numbers, reverts on overflow.
     */
    /*@CTK "SafeMath add"
      @tag spec
      @tag is_pure
      @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)
    */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a);

        return c;
    }

    /**
     * @dev Divides two numbers and returns the remainder (unsigned integer modulo),
     * reverts when dividing by zero.
     */
    /*@CTK "SafeMath mod"
      @tag spec
      @tag is_pure
      @post (b == 0) == __reverted
      @post !__reverted -> __return == a % b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
```

```
     @post !(__has_buf_overflow)
   */
   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
       require(b != 0);
       return a % b;
   }
}
```

File tokens/TRC20/WINK.sol

```
pragma solidity ^0.4.23;

import "./TRC20.sol";
import "../../roles/MinterRole.sol";

/**
 * @title TRC20Detailed token
 * @dev The decimals are only for visualization purposes.
 * All the operations are done using the smallest and indivisible token unit,
 * just as on TRON all the operations are done in sun.
 *
 * Example inherits from basic TRC20 implementation but can be modified to
 * extend from other ITRC20-based tokens:
 * https://github.com/OpenZeppelin/openzeppelin-solidity/issues/1536
 */
contract WINK is TRC20, MinterRole {
   string private _name;
   string private _symbol;
   uint8 private _decimals;

   //@CTK NO_OVERFLOW
   //@CTK NO_BUF_OVERFLOW
   //@CTK NO_ASF
   /*@CTK WINK
     @tag assume_completion
     @post __post._name == name
     @post __post._symbol == symbol
     @post __post._decimals == decimals
   */
   constructor (string name, string symbol, uint8 decimals) public {
       _name = name;
       _symbol = symbol;
       _decimals = decimals;
   }

   /**
    * @return the name of the token.
    */
   //@CTK NO_OVERFLOW
   //@CTK NO_BUF_OVERFLOW
   //@CTK NO_ASF
   /*@CTK name
     @tag assume_completion
     @post __return == _name
   */
   function name() public view returns (string) {
       return _name;
   }

```

```
    /**
     * @return the symbol of the token.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK symbol
      @tag assume_completion
      @post __return == _symbol
    */
    function symbol() public view returns (string) {
        return _symbol;
    }

    /**
     * @return the number of decimals of the token.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK decimals
      @tag assume_completion
      @post __return == _decimals
    */
    function decimals() public view returns (uint8) {
        return _decimals;
    }

    /**
     * @dev Function to mint tokens
     * @param to The address that will receive the minted tokens.
     * @param value The amount of tokens to mint.
     * @return A boolean that indicates if the operation was successful.
     */
    function mint(address to, uint256 value) public onlyMinter returns (bool) {
        _mint(to, value);
        return true;
    }

    /**
     * @dev Burns a specific amount of tokens.
     * @param value The amount of token to be burned.
     */
    function burn(uint256 value) public {
        _burn(msg.sender, value);
    }

    /**
     * @dev Burns a specific amount of tokens from the target address and decrements
       allowance
     * @param from address The address which you want to send tokens from
     * @param value uint256 The amount of token to be burned
     */
    function burnFrom(address from, uint256 value) public {
        _burnFrom(from, value);
    }
}
```

File tokens/TRC20/TRC20.sol

```
pragma solidity ^0.4.23;

import "./ITRC20.sol";
import "../../utils/SafeMath.sol";

/**
 * @title Standard TRC20 token (compatible with ERC20 token)
 *
 * @dev Implementation of the basic standard token.
 * https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
 * Originally based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/
     master/smart_contract/FirstBloodToken.sol
 */
contract TRC20 is ITRC20 {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)) private _allowed;

    uint256 private _totalSupply;

    /**
     * @dev Total number of tokens in existence
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK totalSupply
        @tag assume_completion
        @post (__return) == (_totalSupply)
    */
    function totalSupply() public view returns (uint256) {
        return _totalSupply;
    }

    /**
     * @dev Gets the balance of the specified address.
     * @param owner The address to query the balance of.
     * @return An uint256 representing the amount owned by the passed address.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK balanceOf
      @post __return == __post._balances[owner]
     */
    function balanceOf(address owner) public view returns (uint256) {
        return _balances[owner];
    }

    /**
     * @dev Function to check the amount of tokens that an owner allowed to a spender.
     * @param owner address The address which owns the funds.
     * @param spender address The address which will spend the funds.
     * @return A uint256 specifying the amount of tokens still available for the
         spender.
```

```
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "allowance correctness"
      @post __return == _allowed[owner][spender]
     */
    function allowance(
        address owner,
        address spender
    )
    public
    view
    returns (uint256)
    {
        return _allowed[owner][spender];
    }

    /**
     * @dev Transfer token for a specified address
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@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
     */
    function transfer(address to, uint256 value) public returns (bool) {
        _transfer(msg.sender, to, value);
        return true;
    }

    /**
     * @dev Approve the passed address to spend the specified amount of tokens on behalf of msg.sender.
     * Beware that changing an allowance with this method brings the risk that someone may use both the old
     * and the new allowance by unfortunate transaction ordering. One possible solution to mitigate this
     * race condition is to first reduce the spender's allowance to 0 and set the desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     * @param spender The address which will spend the funds.
     * @param value The amount of tokens to be spent.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "approve normal spender"
      @pre spender != address(0)
      @post __post._allowed[msg.sender][spender] == value
     */
```

```
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "approve zero spender"
      @pre spender == address(0)
      @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender]
     */
    function approve(address spender, uint256 value) public returns (bool) {
        require(spender != address(0));

        _allowed[msg.sender][spender] = value;
        emit Approval(msg.sender, spender, value);
        return true;
    }

    /**
     * @dev Transfer tokens from one address to another
     * @param from address The address which you want to send tokens from
     * @param to address The address which you want to transfer to
     * @param value uint256 the amount of tokens to be transferred
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@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
     */
    function transferFrom(
        address from,
        address to,
        uint256 value
    )
    public
    returns (bool)
    {
        _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
        _transfer(from, to, value);
        return true;
    }

    /**
     * @dev Increase the amount of tokens that an owner allowed to a spender.
     * approve should be called when allowed_[_spender] == 0. To increment
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * @param spender The address which will spend the funds.
     * @param addedValue The amount of tokens to increase the allowance by.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK "increaseAllowance correctness"
      @tag assume_completion
```

```
     @post spender != address(0)
     @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
         addedValue
    */
   function increaseAllowance(
       address spender,
       uint256 addedValue
   )
   public
   returns (bool)
   {
       require(spender != address(0));

       _allowed[msg.sender][spender] = (
       _allowed[msg.sender][spender].add(addedValue));
       emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
       return true;
   }

   /**
    * @dev Decrease the amount of tokens that an owner allowed to a spender.
    * approve should be called when allowed_[_spender] == 0. To decrement
    * allowed value is better to use this function to avoid 2 calls (and wait until
    * the first transaction is mined)
    * From MonolithDAO Token.sol
    * @param spender The address which will spend the funds.
    * @param subtractedValue The amount of tokens to decrease the allowance by.
    */
   //@CTK NO_OVERFLOW
   //@CTK NO_BUF_OVERFLOW
   /*@CTK decreaseAllowance
     @pre _allowed[msg.sender][spender] >= subtractedValue
     @pre spender != address(0)
     @post __post._allowed[msg.sender][spender] ==
         _allowed[msg.sender][spender] - subtractedValue
    */
   function decreaseAllowance(
       address spender,
       uint256 subtractedValue
   )
   public
   returns (bool)
   {
       require(spender != address(0));

       _allowed[msg.sender][spender] = (
       _allowed[msg.sender][spender].sub(subtractedValue));
       emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
       return true;
   }

   /**
    * @dev Transfer token for a specified addresses
    * @param from The address to transfer from.
    * @param to The address to transfer to.
    * @param value The amount to be transferred.
    */
   //@CTK NO_OVERFLOW
```

```
    //@CTK NO_BUF_OVERFLOW
    /*@CTK _transfer_other
      @tag assume_completion
      @pre from != to
      @pre value <= _balances[from]
      @post to != address(0)
      @post __post._balances[from] == _balances[from] - value
      @post __post._balances[to] == _balances[to] + value
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK _transfer_self
      @tag assume_completion
      @pre from == to
      @post to != address(0)
      @post __post._balances[from] == _balances[from]
      @post __post._balances[to] == _balances[to]
     */
    function _transfer(address from, address to, uint256 value) internal {
        require(to != address(0));

        _balances[from] = _balances[from].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(from, to, value);
    }

    /**
     * @dev Internal function that mints an amount of the token and assigns it to
     * an account. This encapsulates the modification of balances such that the
     * proper events are emitted.
     * @param account The account that will receive the created tokens.
     * @param value The amount that will be created.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK mint
      @tag assume_completion
      @post (account != address(0))
      @post (__post._totalSupply) == ((_totalSupply) + (value))
      @post (__post._balances[account]) == ((_balances[account]) + (value))
    */
    function _mint(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.add(value);
        _balances[account] = _balances[account].add(value);
        emit Transfer(address(0), account, value);
    }

    /**
     * @dev Internal function that burns an amount of the token of a given
     * account.
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
```

```
    /*@CTK _burn
      @tag assume_completion
      @post (value <= _balances[account])
      @post (__post._totalSupply) == (_totalSupply - value)
      @post (__post._balances[account]) == (_balances[account] - value)
    */
    function _burn(address account, uint256 value) internal {
        require(account != address(0));

        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
    }

    /**
     * @dev Internal function that burns an amount of the token of a given
     * account, deducting from the sender's allowance for said account. Uses the
     * internal burn function.
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK _burnFrom
      @tag assume_completion
      @post (value <= _allowed[account][msg.sender])
      @post (value <= _balances[account])
      @post (__post._allowed[account][msg.sender]) == (_allowed[account][msg.sender] -
          value)
      @post (__post._balances[account]) == (_balances[account] - value)
      @post __post._totalSupply == (_totalSupply - value)
    */
    function _burnFrom(address account, uint256 value) internal {
        // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be
            accepted,
        // this function needs to emit an event with the updated approval.
        _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
            value);
        _burn(account, value);
    }
}
```