CertiK Audit Report For US Gold

Request Date: 2019-06-28
Revision Date: 2019-06-28
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 US Gold(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 US Gold. This audit was conducted to discover issues and vulnerabilities in the source code of US Gold'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

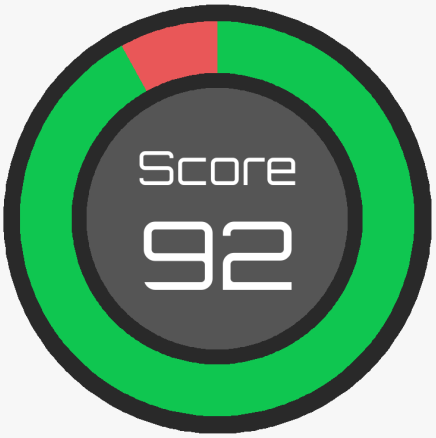
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

Testing Summary

PASS

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

Jun 28, 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.	2	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

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

Vulnerability Details

Critical

No issue found.

Medium

- `WholeIssuableToken`: Possible integer overflow.

Low

- `WholeIssuableToken/issue(uint256 _value, address _target)`: Missing address Check for `_target`.

- `Owned/TransferOwner(address payable newOwner)`: Missing address check for `newOwner`.

(Note: The violations in the formal verification results section are for internal evaluation and are not indication of vulnerabilities in the client code, unless specified in the review section.)

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **DetailedERC20.sol**
  `05dc62fdf8062ece7359be1187da0cfff7976f239a3288aad9c416d28bfaf6fe`
- **ERC20.sol**
  `fb286415cb963f08022b1e4c01a195d6a7ba239e20d20872f3102f1007383471`
- **Owned.sol**
  `54260e967e571ac21e3ee05867450e21c037d1dfb7bdeb61b81db0c0b3cdc2ff`
- **SafeMath.sol**
  `0841465e0699fb9411142199ff4e2e6786a88b17ac8b77cbc3c650ddc599ab80`
- **StandardToken.sol**
  `662f83c127459dd77e3217d49d56073e59a4a3aa96cf916610b111b44824debb`
- **USG.sol**
  `4b781d36ac752b2afb8fe07ac1aae63aaa74e1bae6b27a288375aac19437f41f`
- **WholeIssuableToken.sol**
  `a06079e124e62c83202b4c3f91510433c19a8dc1ecd389c747b1dd2427b86a3f`

Summary

CertiK was chosen by US Gold to audit the design and implementation of its soon to be released USG 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.

USG.sol

- INFO `redeem(uint256, bytes32)` – Recommend using `SafeMath`.
- INFO `StandardToken` – Recommend removing the inheritance of `StandardToken` as it has already been inherited by `WholeIssuableToken`.
- INFO `10**9` – Recommend saving the decimal annotator to a state variable. See below comments in `WholeIssuableToken`.

WholeIssuableToken.sol

- IMPORTANT `mint(uint256, bytes32)`, `issue(uint256, address)` – Possible integer overflow. Recommend using `SafeMath`.
- IMPORTANT `mint(uint256, bytes32)` – Recommend using the `_mint(address, uint256)` method defined in `StandardToken`.
- INFO `10**9` – Recommend saving the decimal annotator to a state variable, which can be used in constructing the `USG` contract.
- INFO `issue(uint256, address)` – Missing address check for `_target`, which may lead to possible value loss due to human error.
- INFO `issue(uint256, address)` – Recommend adding an internal method `_transfer (address sender, address recipient, uint256 amount)` in `StandardToken` which can be used in token issuing.

StandardToken.sol

- INFO `_burnFrom()` – Recommend adding a special event if desired.

Owned.sol

- INFO `TransferOwner()` – Missing address check for `newOwner`, which may lead to possible value loss due to human error. Recommend using the pull model instead of the push model for ownership transfer. Example:

```
address owner;
address proposedOwner;
function proposeNewOwner(address newOwner) isOwner public {
    require(newOwner != address(0), ...);
    proposedOwner = newOwner;
    // emit LogOwnerTransferProposed ...
}
function claimOwnership() public {
    require(msg.sender == proposedOwner, ...);
    owner = proposedOwner;
    proposedOwner = address(0);
    // emit LogOwnerTransferred ...
}
```

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File USG.sol

```
1 pragma solidity ^0.5;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File WholeIssuableToken.sol

```
1 pragma solidity ^0.5;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File Owned.sol

```
1 pragma solidity ^0.5;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File StandardToken.sol

```
1 pragma solidity ^0.5;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

```
1 pragma solidity ^0.5;
```

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

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms

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

CERTIK *label*

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

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

Raw code

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

Counterexample: This code violates the specification

Initial environment

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

If method completes, integer overflow would not happen.

28, Jun 2019
209.26 ms

Line 15 in File USG.sol

```
15    //@CTK NO_OVERFLOW
```

Line 23-28 in File USG.sol

```
23    function redeem(uint256 amt, bytes32 notes) public {
24        uint256 total = amt * 10**9;
25        _burn(msg.sender, total);
26        emit Redeemed(msg.sender, amt, notes);
27
28    }
```

The code meets the specification.

Formal Verification Request 2

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

28, Jun 2019
36.23 ms

Line 16 in File USG.sol

```
16    //@CTK NO_BUF_OVERFLOW
```

Line 23-28 in File USG.sol

```
23    function redeem(uint256 amt, bytes32 notes) public {
24        uint256 total = amt * 10**9;
25        _burn(msg.sender, total);
26        emit Redeemed(msg.sender, amt, notes);
27
28    }
```

The code meets the specification.

Formal Verification Request 3

USG redeem

28, Jun 2019
23.66 ms

Line 17-22 in File USG.sol

```
17    /*@CTK "USG redeem"
18      @tag assume_completion
19      @post ((amt * 1000000000) <= balances_[msg.sender])
20      @post (__post.totalSupply_) == ((totalSupply_) - (amt * 1000000000))
```

```
21      @post (__post.balances_[msg.sender]) == ((balances_[msg.sender]) - (amt *
            1000000000))
22    */
```

Line 23-28 in File USG.sol

```
23    function redeem(uint256 amt, bytes32 notes) public {
24        uint256 total = amt * 10**9;
25        _burn(msg.sender, total);
26        emit Redeemed(msg.sender, amt, notes);
27
28    }
```

The code meets the specification.

Formal Verification Request 4

If method completes, integer overflow would not happen.

28, Jun 2019
26.6 ms

Line 11 in File WholeIssuableToken.sol

```
11    //@CTK NO_OVERFLOW
```

Line 18-27 in File WholeIssuableToken.sol

```
18    function mint(uint256 _value, bytes32 _note) public onlyOwner {
19
20        uint256 totalVal = _value * 10**9;
21
22        balances_[address(this)] += totalVal;
23        totalSupply_ += totalVal;
24        emit Mint(totalVal, _note);
25        emit Transfer(address(0), address(this), totalVal);
26
27    }
```

The code meets the specification.

Formal Verification Request 5

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

28, Jun 2019
0.81 ms

Line 12 in File WholeIssuableToken.sol

```
12    //@CTK NO_BUF_OVERFLOW
```

Line 18-27 in File WholeIssuableToken.sol

```
18    function mint(uint256 _value, bytes32 _note) public onlyOwner {
19
20        uint256 totalVal = _value * 10**9;
21
```

```
22          balances_[address(this)] += totalVal;
23          totalSupply_ += totalVal;
24          emit Mint(totalVal, _note);
25          emit Transfer(address(0), address(this), totalVal);
26
27      }
```

The code meets the specification.

Formal Verification Request 6

WholeIssuableToken mint

28, Jun 2019
22.0 ms

Line 13-17 in File WholeIssuableToken.sol

```
13      /*@CTK "WholeIssuableToken mint"
14        @tag assume_completion
15        @pre (Owner == msg.sender)
16        @post (__post.balances_[address(this)] - balances_[address(this)]) == (__post.
              totalSupply_ - totalSupply_)
17      */
```

Line 18-27 in File WholeIssuableToken.sol

```
18      function mint(uint256 _value, bytes32 _note) public onlyOwner {
19
20          uint256 totalVal = _value * 10**9;
21
22          balances_[address(this)] += totalVal;
23          totalSupply_ += totalVal;
24          emit Mint(totalVal, _note);
25          emit Transfer(address(0), address(this), totalVal);
26
27      }
```

The code meets the specification.

Formal Verification Request 7

If method completes, integer overflow would not happen.

28, Jun 2019
41.41 ms

Line 30 in File WholeIssuableToken.sol

```
30      //@CTK NO_OVERFLOW
```

Line 41-49 in File WholeIssuableToken.sol

```
41      function issue(uint256 _value, address _target) public onlyOwner {
42
43          uint256 totalVal = _value * 10**9;
44
```

```
45        require(balances_[address(this)] >= totalVal);
46        balances_[address(this)] -= totalVal;
47        balances_[_target] += totalVal;
48        emit Transfer(address(this),_target, totalVal);
49    }
```

The code meets the specification.

Formal Verification Request 8

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

28, Jun 2019
1.37 ms

Line 31 in File WholeIssuableToken.sol

```
31    //@CTK NO_BUF_OVERFLOW
```

Line 41-49 in File WholeIssuableToken.sol

```
41    function issue(uint256 _value, address _target) public onlyOwner {
42
43        uint256 totalVal = _value * 10**9;
44
45        require(balances_[address(this)] >= totalVal);
46        balances_[address(this)] -= totalVal;
47        balances_[_target] += totalVal;
48        emit Transfer(address(this),_target, totalVal);
49    }
```

The code meets the specification.

Formal Verification Request 9

WholeIssuableToken issue

28, Jun 2019
123.87 ms

Line 32-40 in File WholeIssuableToken.sol

```
32    /*@CTK "WholeIssuableToken issue"
33      @tag assume_completion
34      @pre (_target != (0))
35      @pre (Owner == msg.sender)
36      @post (balances_[address(this)] >= (_value * 1000000000))
37      @post (__post.balances_[_target] == balances_[_target] + (_value * 1000000000))
38      @post (__post.balances_[address(this)] == balances_[address(this)] - (_value *
            1000000000))
39      @post (__post.balances_[address(this)] - balances_[address(this)]) == (
            totalSupply_ - __post.totalSupply_)
40    */
```

Line 41-49 in File WholeIssuableToken.sol

```
41    function issue(uint256 _value, address _target) public onlyOwner {
42
43        uint256 totalVal = _value * 10**9;
44
45        require(balances_[address(this)] >= totalVal);
46        balances_[address(this)] -= totalVal;
47        balances_[_target] += totalVal;
48        emit Transfer(address(this),_target, totalVal);
49    }
```

The code meets the specification.

Formal Verification Request 10

If method completes, integer overflow would not happen.

28, Jun 2019
5.5 ms

Line 7 in File Owned.sol

```
7   //@CTK NO_OVERFLOW
```

Line 14-17 in File Owned.sol

```
14  constructor() public{
15
16      Owner = msg.sender;
17  }
```

The code meets the specification.

Formal Verification Request 11

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

28, Jun 2019
0.93 ms

Line 8 in File Owned.sol

```
8   //@CTK NO_BUF_OVERFLOW
```

Line 14-17 in File Owned.sol

```
14  constructor() public{
15
16      Owner = msg.sender;
17  }
```

The code meets the specification.

Formal Verification Request 12

Method will not encounter an assertion failure.

28, Jun 2019
0.47 ms

Line 9 in File Owned.sol

```
9  //@CTK NO_ASF
```

Line 14-17 in File Owned.sol

```
14 constructor() public{
15
16     Owner = msg.sender;
17 }
```

The code meets the specification.

Formal Verification Request 13

Owner

28, Jun 2019
0.88 ms

Line 10-13 in File Owned.sol

```
10 /*@CTK Owner
11   @tag assume_completion
12   @post __post.Owner == msg.sender
13  */
```

Line 14-17 in File Owned.sol

```
14 constructor() public{
15
16     Owner = msg.sender;
17 }
```

The code meets the specification.

Formal Verification Request 14

If method completes, integer overflow would not happen.

28, Jun 2019
8.43 ms

Line 19 in File Owned.sol

```
19 //@CTK NO_OVERFLOW
```

Line 26-29 in File Owned.sol

```
26  function IsOwner(address addr) view public returns(bool)
27  {
28      return Owner == addr;
29  }
```

The code meets the specification.

Formal Verification Request 15

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

28, Jun 2019
0.43 ms

Line 20 in File Owned.sol

```
20  //@CTK NO_BUF_OVERFLOW
```

Line 26-29 in File Owned.sol

```
26  function IsOwner(address addr) view public returns(bool)
27  {
28      return Owner == addr;
29  }
```

The code meets the specification.

Formal Verification Request 16

Method will not encounter an assertion failure.

28, Jun 2019
0.52 ms

Line 21 in File Owned.sol

```
21  //@CTK NO_ASF
```

Line 26-29 in File Owned.sol

```
26  function IsOwner(address addr) view public returns(bool)
27  {
28      return Owner == addr;
29  }
```

The code meets the specification.

Formal Verification Request 17

IsOwner

28, Jun 2019
0.44 ms

Line 22-25 in File Owned.sol

```
22  /*@CTK IsOwner
23    @tag assume_completion
24    @post __return == (Owner == addr)
25   */
```

Line 26-29 in File Owned.sol

```
26  function IsOwner(address addr) view public returns(bool)
27  {
28      return Owner == addr;
29  }
```

The code meets the specification.

Formal Verification Request 18

If method completes, integer overflow would not happen.

28, Jun 2019
16.42 ms

Line 31 in File Owned.sol

```
31  //@CTK NO_OVERFLOW
```

Line 41-44 in File Owned.sol

```
41  function TransferOwner(address payable newOwner) public onlyOwner
42  {
43      Owner = newOwner;
44  }
```

The code meets the specification.

Formal Verification Request 19

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

28, Jun 2019
0.73 ms

Line 32 in File Owned.sol

```
32  //@CTK NO_BUF_OVERFLOW
```

Line 41-44 in File Owned.sol

```
41  function TransferOwner(address payable newOwner) public onlyOwner
42  {
43      Owner = newOwner;
44  }
```

The code meets the specification.

Formal Verification Request 20

Method will not encounter an assertion failure.

28, Jun 2019
0.56 ms

Line 33 in File Owned.sol

```
33  //@CTK NO_ASF
```

Line 41-44 in File Owned.sol

```
41  function TransferOwner(address payable newOwner) public onlyOwner
42  {
43      Owner = newOwner;
44  }
```

The code meets the specification.

Formal Verification Request 21

TransferOwner

28, Jun 2019
2.45 ms

Line 34-40 in File Owned.sol

```
34  /*@CTK TransferOwner
35    @tag assume_completion
36    @pre Owner == msg.sender
37    @pre newOwner != address(0)
38    @pre msg.sender != newOwner
39    @post (__post.Owner == newOwner)
40  */
```

Line 41-44 in File Owned.sol

```
41  function TransferOwner(address payable newOwner) public onlyOwner
42  {
43      Owner = newOwner;
44  }
```

The code meets the specification.

Formal Verification Request 22

If method completes, integer overflow would not happen.

28, Jun 2019
6.45 ms

Line 26 in File StandardToken.sol

```
26  //@CTK NO_OVERFLOW
```

Line 33-35 in File StandardToken.sol

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

The code meets the specification.

Formal Verification Request 23

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

28, Jun 2019
0.52 ms

Line 27 in File StandardToken.sol

```
27  //@CTK NO_BUF_OVERFLOW
```

Line 33-35 in File StandardToken.sol

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

The code meets the specification.

Formal Verification Request 24

Method will not encounter an assertion failure.

28, Jun 2019
0.47 ms

Line 28 in File StandardToken.sol

```
28  //@CTK NO_ASF
```

Line 33-35 in File StandardToken.sol

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

The code meets the specification.

Formal Verification Request 25

totalSupply

28, Jun 2019
0.72 ms

Line 29-32 in File StandardToken.sol

```
29  /*@CTK totalSupply
30    @tag assume_completion
31    @post (__return) == (totalSupply_)
32  */
```

Line 33-35 in File StandardToken.sol

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

The code meets the specification.

Formal Verification Request 26

If method completes, integer overflow would not happen.

28, Jun 2019
7.22 ms

Line 42 in File StandardToken.sol

```
42  //@CTK NO_OVERFLOW
```

Line 49-51 in File StandardToken.sol

```
49  function balanceOf(address _owner) public view returns (uint256) {
50    return balances_[_owner];
51  }
```

The code meets the specification.

Formal Verification Request 27

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

28, Jun 2019
0.39 ms

Line 43 in File StandardToken.sol

```
43  //@CTK NO_BUF_OVERFLOW
```

Line 49-51 in File StandardToken.sol

```
49  function balanceOf(address _owner) public view returns (uint256) {
50    return balances_[_owner];
51  }
```

The code meets the specification.

Formal Verification Request 28

Method will not encounter an assertion failure.

28, Jun 2019
0.39 ms

Line 44 in File StandardToken.sol

```
44  //@CTK NO_ASF
```

Line 49-51 in File StandardToken.sol

```
49  function balanceOf(address _owner) public view returns (uint256) {
50    return balances_[_owner];
51  }
```

The code meets the specification.

Formal Verification Request 29

balanceOf

28, Jun 2019
0.39 ms

Line 45-48 in File StandardToken.sol

```
45  /*@CTK balanceOf
46    @tag assume_completion
47    @post (__return) == (balances_[_owner])
48  */
```

Line 49-51 in File StandardToken.sol

```
49  function balanceOf(address _owner) public view returns (uint256) {
50    return balances_[_owner];
51  }
```

The code meets the specification.

Formal Verification Request 30

If method completes, integer overflow would not happen.

28, Jun 2019
8.12 ms

Line 59 in File StandardToken.sol

```
59  //@CTK NO_OVERFLOW
```

Line 66-75 in File StandardToken.sol

```
66  function allowance(
67    address _owner,
68    address _spender
69  )
70  public
71  view
72  returns (uint256)
73  {
74    return allowed_[_owner][_spender];
75  }
```

The code meets the specification.

Formal Verification Request 31

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

28, Jun 2019
0.45 ms

Line 60 in File StandardToken.sol

```
60  //@CTK NO_BUF_OVERFLOW
```

Line 66-75 in File StandardToken.sol

```
66  function allowance(
67    address _owner,
68    address _spender
69  )
70  public
71  view
72  returns (uint256)
73  {
74    return allowed_[_owner][_spender];
75  }
```

The code meets the specification.

Formal Verification Request 32

Method will not encounter an assertion failure.

28, Jun 2019
0.4 ms

Line 61 in File StandardToken.sol

```
61  //@CTK NO_ASF
```

Line 66-75 in File StandardToken.sol

```
66  function allowance(
67    address _owner,
68    address _spender
69  )
70  public
71  view
72  returns (uint256)
73  {
74    return allowed_[_owner][_spender];
75  }
```

The code meets the specification.

Formal Verification Request 33

allowance

28, Jun 2019
0.42 ms

Line 62-65 in File StandardToken.sol

```
62  /*@CTK allowance
63    @tag assume_completion
64    @post (__return) == (allowed_[_owner][_spender])
65  */
```

Line 66-75 in File StandardToken.sol

```
66  function allowance(
67    address _owner,
68    address _spender
69  )
70  public
71  view
72  returns (uint256)
73  {
74    return allowed_[_owner][_spender];
75  }
```

The code meets the specification.

Formal Verification Request 34

If method completes, integer overflow would not happen.

28, Jun 2019
176.84 ms

Line 82 in File StandardToken.sol

```
82  //@CTK NO_OVERFLOW
```

Line 95-103 in File StandardToken.sol

```
95  function transfer(address _to, uint256 _value) public returns (bool) {
96    require(_value <= balances_[msg.sender]);
97    require(_to != address(0));
98
99    balances_[msg.sender] = balances_[msg.sender].sub(_value);
100   balances_[_to] = balances_[_to].add(_value);
101   emit Transfer(msg.sender, _to, _value);
102   return true;
103 }
```

The code meets the specification.

Formal Verification Request 35

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

28, Jun 2019
15.25 ms

Line 83 in File StandardToken.sol

```
83  //@CTK NO_BUF_OVERFLOW
```

Line 95-103 in File StandardToken.sol

```
95  function transfer(address _to, uint256 _value) public returns (bool) {
96    require(_value <= balances_[msg.sender]);
97    require(_to != address(0));
98
99    balances_[msg.sender] = balances_[msg.sender].sub(_value);
100   balances_[_to] = balances_[_to].add(_value);
101   emit Transfer(msg.sender, _to, _value);
102   return true;
103 }
```

The code meets the specification.

Formal Verification Request 36

Method will not encounter an assertion failure.

28, Jun 2019
71.31 ms

Line 84 in File StandardToken.sol

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

Line 95-103 in File StandardToken.sol

```
95  function transfer(address _to, uint256 _value) public returns (bool) {
96    require(_value <= balances_[msg.sender]);
97    require(_to != address(0));
98
99    balances_[msg.sender] = balances_[msg.sender].sub(_value);
100   balances_[_to] = balances_[_to].add(_value);
101   emit Transfer(msg.sender, _to, _value);
102   return true;
103 }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3     Input = {
4         _to = 8
5         _value = 118
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        }
```

```
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "StandardToken",
          "balance": 0,
          "contract": {
            "balances_": [
              {
                "key": 8,
                "value": 138
              },
              {
                "key": "ALL_OTHERS",
                "value": 134
              }
            ],
            "allowed_": [
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 134
                  }
                ]
              }
            ],
            "totalSupply_": 0
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 37

transfer

📅 28, Jun 2019
⏱ 211.12 ms

Line 85-94 in File StandardToken.sol

```
85  /*@CTK transfer
86    @tag assume_completion
87    @pre _to != address(0)
88    @pre _value <= balances_[msg.sender]
89    @post (msg.sender != _to) -> (__post.balances_[_to] == balances_[_to] + _value)
90    @post (msg.sender != _to) -> (__post.balances_[msg.sender] == balances_[msg.sender
          ] - _value)
91    @post (msg.sender == _to) -> (__post.balances_[_to] == balances_[_to])
92    @post (msg.sender == _to) -> (__post.balances_[msg.sender] == balances_[msg.sender
          ])
93    @post __return == true
94  */
```

Line 95-103 in File StandardToken.sol

```
95  function transfer(address _to, uint256 _value) public returns (bool) {
96    require(_value <= balances_[msg.sender]);
97    require(_to != address(0));
98
99    balances_[msg.sender] = balances_[msg.sender].sub(_value);
100   balances_[_to] = balances_[_to].add(_value);
101   emit Transfer(msg.sender, _to, _value);
102   return true;
103 }
```

The code meets the specification.

Formal Verification Request 38

If method completes, integer overflow would not happen.

28, Jun 2019
10.24 ms

Line 114 in File StandardToken.sol

```
114 //@CTK NO_OVERFLOW
```

Line 121-125 in File StandardToken.sol

```
121 function approve(address _spender, uint256 _value) public returns (bool) {
122   allowed_[msg.sender][_spender] = _value;
123   emit Approval(msg.sender, _spender, _value);
124   return true;
125 }
```

The code meets the specification.

Formal Verification Request 39

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

28, Jun 2019
0.44 ms

Line 115 in File StandardToken.sol

```
115  //@CTK NO_BUF_OVERFLOW
```

Line 121-125 in File StandardToken.sol

```
121  function approve(address _spender, uint256 _value) public returns (bool) {
122    allowed_[msg.sender][_spender] = _value;
123    emit Approval(msg.sender, _spender, _value);
124    return true;
125  }
```

The code meets the specification.

Formal Verification Request 40

Method will not encounter an assertion failure.

28, Jun 2019
0.5 ms

Line 116 in File StandardToken.sol

```
116  //@CTK NO_ASF
```

Line 121-125 in File StandardToken.sol

```
121  function approve(address _spender, uint256 _value) public returns (bool) {
122    allowed_[msg.sender][_spender] = _value;
123    emit Approval(msg.sender, _spender, _value);
124    return true;
125  }
```

The code meets the specification.

Formal Verification Request 41

approve

28, Jun 2019
1.5 ms

Line 117-120 in File StandardToken.sol

```
117  /*@CTK approve
118    @tag assume_completion
119    @post (__post.allowed_[msg.sender][_spender]) == (_value)
120  */
```

Line 121-125 in File StandardToken.sol

```
121  function approve(address _spender, uint256 _value) public returns (bool) {
122    allowed_[msg.sender][_spender] = _value;
123    emit Approval(msg.sender, _spender, _value);
124    return true;
125  }
```

The code meets the specification.

Formal Verification Request 42

If method completes, integer overflow would not happen.

28, Jun 2019
194.86 ms

Line 133 in File StandardToken.sol

```
133  //@CTK NO_OVERFLOW
```

Line 148-165 in File StandardToken.sol

```
148  function transferFrom(
149    address _from,
150    address _to,
151    uint256 _value
152  )
153  public
154  returns (bool)
155  {
156    require(_value <= balances_[_from]);
157    require(_value <= allowed_[_from][msg.sender]);
158    require(_to != address(0));
159
160    balances_[_from] = balances_[_from].sub(_value);
161    balances_[_to] = balances_[_to].add(_value);
162    allowed_[_from][msg.sender] = allowed_[_from][msg.sender].sub(_value);
163    emit Transfer(_from, _to, _value);
164    return true;
165  }
```

The code meets the specification.

Formal Verification Request 43

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

28, Jun 2019
15.71 ms

Line 134 in File StandardToken.sol

```
134  //@CTK NO_BUF_OVERFLOW
```

Line 148-165 in File StandardToken.sol

```
148  function transferFrom(
149    address _from,
150    address _to,
151    uint256 _value
152  )
153  public
154  returns (bool)
155  {
156    require(_value <= balances_[_from]);
157    require(_value <= allowed_[_from][msg.sender]);
158    require(_to != address(0));
159
```

```
160     balances_[_from] = balances_[_from].sub(_value);
161     balances_[_to] = balances_[_to].add(_value);
162     allowed_[_from][msg.sender] = allowed_[_from][msg.sender].sub(_value);
163     emit Transfer(_from, _to, _value);
164     return true;
165   }
```

The code meets the specification.

Formal Verification Request 44

Method will not encounter an assertion failure.

28, Jun 2019
140.08 ms

Line 135 in File StandardToken.sol

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

Line 148-165 in File StandardToken.sol

```
148   function transferFrom(
149     address _from,
150     address _to,
151     uint256 _value
152   )
153   public
154   returns (bool)
155   {
156     require(_value <= balances_[_from]);
157     require(_value <= allowed_[_from][msg.sender]);
158     require(_to != address(0));
159
160     balances_[_from] = balances_[_from].sub(_value);
161     balances_[_to] = balances_[_to].add(_value);
162     allowed_[_from][msg.sender] = allowed_[_from][msg.sender].sub(_value);
163     emit Transfer(_from, _to, _value);
164     return true;
165   }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3     Input = {
4         _from = 0
5         _to = 2
6         _value = 128
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 = {
```

```
        "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": {
            "balances_": [
              {
                "key": 64,
                "value": 0
              },
              {
                "key": 128,
                "value": 0
              },
              {
                "key": 1,
                "value": 0
              },
              {
                "key": 4,
                "value": 0
              },
              {
                "key": 2,
                "value": 224
              },
              {
                "key": 16,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 128
              }
            ],
            "allowed_": [
              {
                "key": 0,
                "value": [
                  {
                    "key": 8,
                    "value": 4
                  },
                  {
```

```
                  "key": 2,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```


Formal Verification Request 45

transferFrom

28, Jun 2019
438.78 ms

Line 136-147 in File StandardToken.sol

```
/*@CTK "transferFrom"
  @tag assume_completion
  @pre (_to) != (address(0))
  @pre (_value) <= (balances_[_from])
  @pre (_value) <= (allowed_[_from][msg.sender])
  @post (_from != _to) -> (__post.balances_[_to] == (balances_[_to] + _value))
  @post (_from != _to) -> (__post.balances_[_from] == (balances_[_from] - _value))
  @post (_from == _to) -> (__post.balances_[_to] == balances_[_to])
  @post (_from == _to) -> (__post.balances_[_from] == balances_[_from])
  @post (__post.allowed_[_from][msg.sender]) == (allowed_[_from][msg.sender] -
      _value)
  @post (__return) == (true)
*/
```

Line 148-165 in File StandardToken.sol

```
function transferFrom(
  address _from,
```

```
150     address _to,
151     uint256 _value
152   )
153   public
154   returns (bool)
155   {
156     require(_value <= balances_[_from]);
157     require(_value <= allowed_[_from][msg.sender]);
158     require(_to != address(0));
159
160     balances_[_from] = balances_[_from].sub(_value);
161     balances_[_to] = balances_[_to].add(_value);
162     allowed_[_from][msg.sender] = allowed_[_from][msg.sender].sub(_value);
163     emit Transfer(_from, _to, _value);
164     return true;
165   }
```

The code meets the specification.

Formal Verification Request 46

If method completes, integer overflow would not happen.

28, Jun 2019
40.48 ms

Line 176 in File StandardToken.sol

```
176   //@CTK NO_OVERFLOW
```

Line 186-197 in File StandardToken.sol

```
186   function increaseApproval(
187     address _spender,
188     uint256 _addedValue
189   )
190   public
191   returns (bool)
192   {
193     allowed_[msg.sender][_spender] = (
194     allowed_[msg.sender][_spender].add(_addedValue));
195     emit Approval(msg.sender, _spender, allowed_[msg.sender][_spender]);
196     return true;
197   }
```

The code meets the specification.

Formal Verification Request 47

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

28, Jun 2019
0.85 ms

Line 177 in File StandardToken.sol

```
177   //@CTK NO_BUF_OVERFLOW
```

Line 186-197 in File StandardToken.sol

```
186   function increaseApproval(
187     address _spender,
188     uint256 _addedValue
189   )
190   public
191   returns (bool)
192   {
193     allowed_[msg.sender][_spender] = (
194     allowed_[msg.sender][_spender].add(_addedValue));
195     emit Approval(msg.sender, _spender, allowed_[msg.sender][_spender]);
196     return true;
197   }
```

✅ The code meets the specification.

Formal Verification Request 48

Method will not encounter an assertion failure.

📅 28, Jun 2019
⏱ 7.83 ms

Line 178 in File StandardToken.sol

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

Line 186-197 in File StandardToken.sol

```
186   function increaseApproval(
187     address _spender,
188     uint256 _addedValue
189   )
190   public
191   returns (bool)
192   {
193     allowed_[msg.sender][_spender] = (
194     allowed_[msg.sender][_spender].add(_addedValue));
195     emit Approval(msg.sender, _spender, allowed_[msg.sender][_spender]);
196     return true;
197   }
```

❌ This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3     Input = {
4         _addedValue = 241
5         _spender = 0
6     }
7     This = 0
8     Internal = {
9         __has_assertion_failure = false
10        __has_buf_overflow = false
11        __has_overflow = false
12        __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": {
            "balances_": [
              {
                "key": 0,
                "value": 2
              },
              {
                "key": "ALL_OTHERS",
                "value": 241
              }
            ],
            "allowed_": [
              {
                "key": 0,
                "value": [
                  {
                    "key": 0,
                    "value": 111
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 241
                  }
                ]
              },
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 241
                  }
                ]
              }
            ],
            "totalSupply_": 0
          }
        }
```

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

Function invocation is reverted.
```

Formal Verification Request 49

increaseApproval

📅 28, Jun 2019
⏱ 3.89 ms

Line 179-185 in File StandardToken.sol

```
/*@CTK increaseApproval
  @tag assume_completion
  @pre _spender != 0x0
  @pre _spender != msg.sender
  @post (__post.allowed_[msg.sender][_spender]) == (allowed_[msg.sender][_spender] +
      _addedValue)
  @post (__return) == (true)
*/
```

Line 186-197 in File StandardToken.sol

```
function increaseApproval(
  address _spender,
  uint256 _addedValue
)
public
returns (bool)
{
  allowed_[msg.sender][_spender] = (
  allowed_[msg.sender][_spender].add(_addedValue));
  emit Approval(msg.sender, _spender, allowed_[msg.sender][_spender]);
  return true;
}
```

✅ The code meets the specification.

Formal Verification Request 50

If method completes, integer overflow would not happen.

📅 28, Jun 2019
⏱ 55.35 ms

Line 208 in File StandardToken.sol

```
//@CTK NO_OVERFLOW
```

Line 218-233 in File StandardToken.sol

```
218  function decreaseApproval(
219    address _spender,
220    uint256 _subtractedValue
221  )
222  public
223  returns (bool)
224  {
225    uint256 oldValue = allowed_[msg.sender][_spender];
226    if (_subtractedValue >= oldValue) {
227      allowed_[msg.sender][_spender] = 0;
228    } else {
229      allowed_[msg.sender][_spender] = oldValue.sub(_subtractedValue);
230    }
231    emit Approval(msg.sender, _spender, allowed_[msg.sender][_spender]);
232    return true;
233  }
```

The code meets the specification.

Formal Verification Request 51

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

28, Jun 2019
1.62 ms

Line 209 in File StandardToken.sol

```
209  //@CTK NO_BUF_OVERFLOW
```

Line 218-233 in File StandardToken.sol

```
218  function decreaseApproval(
219    address _spender,
220    uint256 _subtractedValue
221  )
222  public
223  returns (bool)
224  {
225    uint256 oldValue = allowed_[msg.sender][_spender];
226    if (_subtractedValue >= oldValue) {
227      allowed_[msg.sender][_spender] = 0;
228    } else {
229      allowed_[msg.sender][_spender] = oldValue.sub(_subtractedValue);
230    }
231    emit Approval(msg.sender, _spender, allowed_[msg.sender][_spender]);
232    return true;
233  }
```

The code meets the specification.

Formal Verification Request 52

Method will not encounter an assertion failure.

28, Jun 2019
6.6 ms

Line 210 in File StandardToken.sol

```
210  //@CTK NO_ASF
```

Line 218-233 in File StandardToken.sol

```
218  function decreaseApproval(
219    address _spender,
220    uint256 _subtractedValue
221  )
222  public
223  returns (bool)
224  {
225    uint256 oldValue = allowed_[msg.sender][_spender];
226    if (_subtractedValue >= oldValue) {
227      allowed_[msg.sender][_spender] = 0;
228    } else {
229      allowed_[msg.sender][_spender] = oldValue.sub(_subtractedValue);
230    }
231    emit Approval(msg.sender, _spender, allowed_[msg.sender][_spender]);
232    return true;
233  }
```

The code meets the specification.

Formal Verification Request 53

decreaseApproval

28, Jun 2019
44.7 ms

Line 211-217 in File StandardToken.sol

```
211  /*@CTK decreaseApproval
212    @tag assume_completion
213    @pre _spender != 0x0
214    @pre _spender != msg.sender
215    @post (_subtractedValue > allowed_[msg.sender][_spender]) -> (__post.allowed_[msg.
           sender][_spender] == 0)
216    @post (_subtractedValue <= allowed_[msg.sender][_spender]) -> (__post.allowed_[msg
           .sender][_spender] == allowed_[msg.sender][_spender] - _subtractedValue)
217  */
```

Line 218-233 in File StandardToken.sol

```
218  function decreaseApproval(
219    address _spender,
220    uint256 _subtractedValue
221  )
222  public
223  returns (bool)
224  {
225    uint256 oldValue = allowed_[msg.sender][_spender];
226    if (_subtractedValue >= oldValue) {
227      allowed_[msg.sender][_spender] = 0;
228    } else {
229      allowed_[msg.sender][_spender] = oldValue.sub(_subtractedValue);
```

```
230     }
231     emit Approval(msg.sender, _spender, allowed_[msg.sender][_spender]);
232     return true;
233   }
```

The code meets the specification.

Formal Verification Request 54

If method completes, integer overflow would not happen.

28, Jun 2019
110.51 ms

Line 242 in File StandardToken.sol

```
242   //@CTK NO_OVERFLOW
```

Line 251-256 in File StandardToken.sol

```
251   function _mint(address _account, uint256 _amount) internal {
252     require(_account != address(0));
253     totalSupply_ = totalSupply_.add(_amount);
254     balances_[_account] = balances_[_account].add(_amount);
255     emit Transfer(address(0), _account, _amount);
256   }
```

The code meets the specification.

Formal Verification Request 55

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

28, Jun 2019
14.88 ms

Line 243 in File StandardToken.sol

```
243   //@CTK NO_BUF_OVERFLOW
```

Line 251-256 in File StandardToken.sol

```
251   function _mint(address _account, uint256 _amount) internal {
252     require(_account != address(0));
253     totalSupply_ = totalSupply_.add(_amount);
254     balances_[_account] = balances_[_account].add(_amount);
255     emit Transfer(address(0), _account, _amount);
256   }
```

The code meets the specification.

Formal Verification Request 56

Method will not encounter an assertion failure.

28, Jun 2019
53.19 ms

Line 244 in File StandardToken.sol

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

Line 251-256 in File StandardToken.sol

```
251   function _mint(address _account, uint256 _amount) internal {
252     require(_account != address(0));
253     totalSupply_ = totalSupply_.add(_amount);
254     balances_[_account] = balances_[_account].add(_amount);
255     emit Transfer(address(0), _account, _amount);
256   }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3     Input = {
4         _account = 1
5         _amount = 128
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        block = {
22          "number": 0,
23          "timestamp": 0
24        }
25    }
26    Address_Map = [
27      {
28        "key": 0,
29        "value": {
30          "contract_name": "StandardToken",
31          "balance": 0,
32          "contract": {
33            "balances_": [
34              {
35                "key": 1,
36                "value": 128
37              },
38              {
```

```
              "key": "ALL_OTHERS",
              "value": 1
            }
          ],
          "allowed_": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 1
                }
              ]
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 57

mint

28, Jun 2019
40.15 ms

Line 245-250 in File StandardToken.sol

```
/*@CTK mint
  @tag assume_completion
  @pre (_account != address(0))
  @post (__post.totalSupply_) == ((totalSupply_) + (_amount))
  @post (__post.balances_[_account]) == ((balances_[_account]) + (_amount))
*/
```

Line 251-256 in File StandardToken.sol

```
function _mint(address _account, uint256 _amount) internal {
  require(_account != address(0));
  totalSupply_ = totalSupply_.add(_amount);
  balances_[_account] = balances_[_account].add(_amount);
  emit Transfer(address(0), _account, _amount);
}
```

The code meets the specification.

Formal Verification Request 58

If method completes, integer overflow would not happen.

28, Jun 2019
97.79 ms

Line 264 in File StandardToken.sol

```
264  //@CTK NO_OVERFLOW
```

Line 274-281 in File StandardToken.sol

```
274  function _burn(address _account, uint256 _amount) internal {
275    require(_account != address(0));
276    require(_amount <= balances_[_account]);
277
278    totalSupply_ = totalSupply_.sub(_amount);
279    balances_[_account] = balances_[_account].sub(_amount);
280    emit Transfer(_account, address(0), _amount);
281  }
```

The code meets the specification.

Formal Verification Request 59

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

28, Jun 2019
27.77 ms

Line 265 in File StandardToken.sol

```
265  //@CTK NO_BUF_OVERFLOW
```

Line 274-281 in File StandardToken.sol

```
274  function _burn(address _account, uint256 _amount) internal {
275    require(_account != address(0));
276    require(_amount <= balances_[_account]);
277
278    totalSupply_ = totalSupply_.sub(_amount);
279    balances_[_account] = balances_[_account].sub(_amount);
280    emit Transfer(_account, address(0), _amount);
281  }
```

The code meets the specification.

Formal Verification Request 60

Method will not encounter an assertion failure.

28, Jun 2019
58.58 ms

Line 266 in File StandardToken.sol

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

Line 274-281 in File StandardToken.sol

```
274   function _burn(address _account, uint256 _amount) internal {
275     require(_account != address(0));
276     require(_amount <= balances_[_account]);
277
278     totalSupply_ = totalSupply_.sub(_amount);
279     balances_[_account] = balances_[_account].sub(_amount);
280     emit Transfer(_account, address(0), _amount);
281   }
```

This code violates the specification.

```
1   Counter Example:
2   Before Execution:
3       Input = {
4           _account = 4
5           _amount = 66
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          block = {
22            "number": 0,
23            "timestamp": 0
24          }
25      }
26      Address_Map = [
27        {
28          "key": 0,
29          "value": {
30            "contract_name": "StandardToken",
31            "balance": 0,
32            "contract": {
33              "balances_": [
34                {
35                  "key": 4,
36                  "value": 132
37                },
38                {
39                  "key": 0,
40                  "value": 0
41                },
42                {
43                  "key": 32,
44                  "value": 1
45                },
```

```
            {
              "key": "ALL_OTHERS",
              "value": 66
            }
          ],
          "allowed_": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 32
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            },
            {
              "key": 32,
              "value": [
                {
                  "key": 0,
                  "value": 128
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 66
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 66
                }
              ]
            }
          ],
          "totalSupply_": 2
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 61

burn

28, Jun 2019
178.22 ms

Line 267-273 in File StandardToken.sol

```
267  /*@CTK burn
268    @tag assume_completion
269    @pre (_account != address(0))
270    @post (_amount <= balances_[_account])
271    @post (__post.totalSupply_) == ((totalSupply_) - (_amount))
272    @post (__post.balances_[_account]) == ((balances_[_account]) - (_amount))
273  */
```

Line 274-281 in File StandardToken.sol

```
274  function _burn(address _account, uint256 _amount) internal {
275    require(_account != address(0));
276    require(_amount <= balances_[_account]);
277
278    totalSupply_ = totalSupply_.sub(_amount);
279    balances_[_account] = balances_[_account].sub(_amount);
280    emit Transfer(_account, address(0), _amount);
281  }
```

The code meets the specification.

Formal Verification Request 62

If method completes, integer overflow would not happen.

28, Jun 2019
143.63 ms

Line 290 in File StandardToken.sol

```
290  //@CTK NO_OVERFLOW
```

Line 300-308 in File StandardToken.sol

```
300  function _burnFrom(address _account, uint256 _amount) internal {
301    require(_amount <= allowed_[_account][msg.sender]);
302
303    // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
       ,
304    // this function needs to emit an event with the updated approval.
305    allowed_[_account][msg.sender] = allowed_[_account][msg.sender].sub(
306      _amount);
307    _burn(_account, _amount);
308  }
```

The code meets the specification.

Formal Verification Request 63

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

28, Jun 2019
22.11 ms

Line 291 in File StandardToken.sol

```
291  //@CTK NO_BUF_OVERFLOW
```

Line 300-308 in File StandardToken.sol

```
300  function _burnFrom(address _account, uint256 _amount) internal {
301    require(_amount <= allowed_[_account][msg.sender]);
302
303    // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
       ,
304    // this function needs to emit an event with the updated approval.
305    allowed_[_account][msg.sender] = allowed_[_account][msg.sender].sub(
306      _amount);
307    _burn(_account, _amount);
308  }
```

The code meets the specification.

Formal Verification Request 64

Method will not encounter an assertion failure.

28, Jun 2019
84.3 ms

Line 292 in File StandardToken.sol

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

Line 300-308 in File StandardToken.sol

```
300  function _burnFrom(address _account, uint256 _amount) internal {
301    require(_amount <= allowed_[_account][msg.sender]);
302
303    // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
       ,
304    // this function needs to emit an event with the updated approval.
305    allowed_[_account][msg.sender] = allowed_[_account][msg.sender].sub(
306      _amount);
307    _burn(_account, _amount);
308  }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3     Input = {
4         _account = 1
5         _amount = 1
6     }
7     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 = {
    block = {
      "number": 0,
      "timestamp": 0
    }
}
Address_Map = [
  {
    "key": 0,
    "value": {
      "contract_name": "StandardToken",
      "balance": 0,
      "contract": {
        "balances_": [
          {
            "key": 4,
            "value": 32
          },
          {
            "key": 0,
            "value": 0
          },
          {
            "key": 128,
            "value": 0
          },
          {
            "key": "ALL_OTHERS",
            "value": 128
          }
        ],
        "allowed_": [
          {
            "key": 2,
            "value": [
              {
                "key": 0,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 128
              }
            ]
          },
          {
```

```
            "key": 1,
            "value": [
              {
                "key": 4,
                "value": 0
              },
              {
                "key": 0,
                "value": 128
              },
              {
                "key": 16,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 1
              }
            ]
          },
          {
            "key": 0,
            "value": [
              {
                "key": 0,
                "value": 1
              },
              {
                "key": 16,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 16
              }
            ]
          },
          {
            "key": 128,
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": 128
              }
            ]
          },
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": 128
              }
            ]
          }
        ],
        "totalSupply_": 0
      }
```

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

Function invocation is reverted.
```

Formal Verification Request 65

burnFrom

28, Jun 2019
70.34 ms

Line 293-299 in File StandardToken.sol

```
/*@CTK burnFrom
  @tag assume_completion
  @post (_amount <= allowed_[_account][msg.sender])
  @post (_amount <= balances_[_account])
  @post (__post.allowed_[_account][msg.sender]) == ((allowed_[_account][msg.sender])
      - (_amount))
  @post (__post.balances_[_account]) == ((balances_[_account]) - (_amount))
*/
```

Line 300-308 in File StandardToken.sol

```
function _burnFrom(address _account, uint256 _amount) internal {
  require(_amount <= allowed_[_account][msg.sender]);

  // 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(
    _amount);
  _burn(_account, _amount);
}
```

The code meets the specification.

Formal Verification Request 66

Method will not encounter an assertion failure.

28, Jun 2019
34.27 ms

Line 13 in File SafeMath.sol

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

Line 21-28 in File SafeMath.sol

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

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 67

SafeMath mul

28, Jun 2019
419.43 ms

Line 14-20 in File SafeMath.sol

```
14  /*@CTK "SafeMath mul"
15    @post ((a > 0) && (((a * b) / a) != b)) == (__reverted)
16    @post !__reverted -> __return == a * b
17    @post !__reverted == !__has_overflow
```

```
18    @post !__reverted -> !(__has_assertion_failure)
19    @post !(__has_buf_overflow)
20    */
```

Line 21-28 in File SafeMath.sol

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

The code meets the specification.

Formal Verification Request 68

Method will not encounter an assertion failure.

28, Jun 2019
9.32 ms

Line 33 in File SafeMath.sol

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

Line 41-46 in File SafeMath.sol

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

This code violates the specification.

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

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

Function invocation is reverted.
```

Formal Verification Request 69

SafeMath div

28, Jun 2019
1.2 ms

Line 34-40 in File SafeMath.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 41-46 in File SafeMath.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 a / b;
}
```

The code meets the specification.

Formal Verification Request 70

Method will not encounter an assertion failure.

28, Jun 2019
16.85 ms

Line 51 in File SafeMath.sol

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

Line 59-62 in File SafeMath.sol

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

```
61      return a - b;
62    }
```

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 71

SafeMath sub

28, Jun 2019
1.41 ms

Line 52-58 in File SafeMath.sol

```
52  /*@CTK "SafeMath sub"
53    @post (a < b) == __reverted
54    @post !__reverted -> __return == a - b
55    @post !__reverted -> !__has_overflow
56    @post !__reverted -> !(__has_assertion_failure)
57    @post !(__has_buf_overflow)
58  */
```

Line 59-62 in File SafeMath.sol

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

The code meets the specification.

Formal Verification Request 72

Method will not encounter an assertion failure.

28, Jun 2019
17.1 ms

Line 67 in File SafeMath.sol

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

Line 75-79 in File SafeMath.sol

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

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        a = 3
        b = 253
    }
    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"
      }
```

```
31     ]
32
33 Function invocation is reverted.
```

Formal Verification Request 73

SafeMath add

28, Jun 2019
3.01 ms

Line 68-74 in File SafeMath.sol

```
68  /*@CTK "SafeMath add"
69    @post (a + b < a || a + b < b) == __reverted
70    @post !__reverted -> __return == a + b
71    @post !__reverted -> !__has_overflow
72    @post !__reverted -> !(__has_assertion_failure)
73    @post !(__has_buf_overflow)
74  */
```

Line 75-79 in File SafeMath.sol

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

The code meets the specification.

Source Code with CertiK Labels

File USG.sol

```
pragma solidity ^0.5;

import "./StandardToken.sol";
import "./DetailedERC20.sol";
import "./SafeMath.sol";
import "./WholeIssuableToken.sol";
import "./Owned.sol";

contract USG is StandardToken, DetailedERC20, WholeIssuableToken {
    constructor() DetailedERC20("USGold", "USG", 9) public {}

    event Redeemed(address addr, uint256 amt, bytes32 notes);

    //mut be whole token
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK "USG redeem"
      @tag assume_completion
      @post ((amt * 1000000000) <= balances_[msg.sender])
      @post (__post.totalSupply_) == ((totalSupply_) - (amt * 1000000000))
      @post (__post.balances_[msg.sender]) == ((balances_[msg.sender]) - (amt *
          1000000000))
    */
    function redeem(uint256 amt, bytes32 notes) public {
        uint256 total = amt * 10**9;
        _burn(msg.sender, total);
        emit Redeemed(msg.sender, amt, notes);

    }

}
```

File WholeIssuableToken.sol

```
pragma solidity ^0.5;

import "./Owned.sol";
import "./StandardToken.sol";

contract WholeIssuableToken is StandardToken, Owned {

    event Mint(uint256 indexed _value, bytes32 indexed _note);

    /*_value is WHOLE tokens*/
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    /*@CTK "WholeIssuableToken mint"
      @tag assume_completion
      @pre (Owner == msg.sender)
      @post (__post.balances_[address(this)] - balances_[address(this)]) == (__post.
          totalSupply_ - totalSupply_)
    */
    function mint(uint256 _value, bytes32 _note) public onlyOwner {

        uint256 totalVal = _value * 10**9;
```

```
21
22          balances_[address(this)] += totalVal;
23          totalSupply_ += totalVal;
24          emit Mint(totalVal, _note);
25          emit Transfer(address(0), address(this), totalVal);
26
27      }
28
29      /*_value is WHOLE tokens*/
30      //@CTK NO_OVERFLOW
31      //@CTK NO_BUF_OVERFLOW
32      /*@CTK "WholeIssuableToken issue"
33        @tag assume_completion
34        @pre (_target != (0))
35        @pre (Owner == msg.sender)
36        @post (balances_[address(this)] >= (_value * 1000000000))
37        @post (__post.balances_[_target] == balances_[_target] + (_value * 1000000000))
38        @post (__post.balances_[address(this)] == balances_[address(this)] - (_value *
              1000000000))
39        @post (__post.balances_[address(this)] - balances_[address(this)]) == (
              totalSupply_ - __post.totalSupply_)
40      */
41      function issue(uint256 _value, address _target) public onlyOwner {
42
43          uint256 totalVal = _value * 10**9;
44
45          require(balances_[address(this)] >= totalVal);
46          balances_[address(this)] -= totalVal;
47          balances_[_target] += totalVal;
48          emit Transfer(address(this),_target, totalVal);
49      }
50  }
```

File Owned.sol

```
1   pragma solidity ^0.5;
2
3   contract Owned {
4     //address payable private Owner;
5     address payable internal Owner;
6
7     //@CTK NO_OVERFLOW
8     //@CTK NO_BUF_OVERFLOW
9     //@CTK NO_ASF
10    /*@CTK Owner
11      @tag assume_completion
12      @post __post.Owner == msg.sender
13     */
14    constructor() public{
15
16        Owner = msg.sender;
17    }
18
19    //@CTK NO_OVERFLOW
20    //@CTK NO_BUF_OVERFLOW
21    //@CTK NO_ASF
22    /*@CTK IsOwner
23      @tag assume_completion
24      @post __return == (Owner == addr)
```

```
   */
  function IsOwner(address addr) view public returns(bool)
  {
      return Owner == addr;
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK TransferOwner
    @tag assume_completion
    @pre Owner == msg.sender
    @pre newOwner != address(0)
    @pre msg.sender != newOwner
    @post (__post.Owner == newOwner)
  */
  function TransferOwner(address payable newOwner) public onlyOwner
  {
      Owner = newOwner;
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK TransferOwner
    @pre Owner == msg.sender
  */
  function Terminate() public onlyOwner
  {
      selfdestruct(Owner);
  }

  modifier onlyOwner(){
        require(msg.sender == Owner);
        _;
    }
}
```

File StandardToken.sol

```
pragma solidity ^0.5;

import "./ERC20.sol";
import "./SafeMath.sol";


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

  mapping (address => uint256) public balances_;

```

```
19  mapping (address => mapping (address => uint256)) public allowed_;
20
21  uint256 public totalSupply_;
22
23  /**
24  * @dev Total number of tokens in existence
25  */
26  //@CTK NO_OVERFLOW
27  //@CTK NO_BUF_OVERFLOW
28  //@CTK NO_ASF
29  /*@CTK totalSupply
30    @tag assume_completion
31    @post (__return) == (totalSupply_)
32  */
33  function totalSupply() public view returns (uint256) {
34    return totalSupply_;
35  }
36
37  /**
38  * @dev Gets the balance of the specified address.
39  * @param _owner The address to query the the balance of.
40  * @return An uint256 representing the amount owned by the passed address.
41  */
42  //@CTK NO_OVERFLOW
43  //@CTK NO_BUF_OVERFLOW
44  //@CTK NO_ASF
45  /*@CTK balanceOf
46    @tag assume_completion
47    @post (__return) == (balances_[_owner])
48  */
49  function balanceOf(address _owner) public view returns (uint256) {
50    return balances_[_owner];
51  }
52
53  /**
54   * @dev Function to check the amount of tokens that an owner allowed to a spender.
55   * @param _owner address The address which owns the funds.
56   * @param _spender address The address which will spend the funds.
57   * @return A uint256 specifying the amount of tokens still available for the spender
         .
58   */
59  //@CTK NO_OVERFLOW
60  //@CTK NO_BUF_OVERFLOW
61  //@CTK NO_ASF
62  /*@CTK allowance
63    @tag assume_completion
64    @post (__return) == (allowed_[_owner][_spender])
65  */
66  function allowance(
67    address _owner,
68    address _spender
69  )
70  public
71  view
72  returns (uint256)
73  {
74    return allowed_[_owner][_spender];
75  }
```

```
76
77  /**
78  * @dev Transfer token for a specified address
79  * @param _to The address to transfer to.
80  * @param _value The amount to be transferred.
81  */
82  //@CTK NO_OVERFLOW
83  //@CTK NO_BUF_OVERFLOW
84  //@CTK FAIL NO_ASF
85  /*@CTK transfer
86    @tag assume_completion
87    @pre _to != address(0)
88    @pre _value <= balances_[msg.sender]
89    @post (msg.sender != _to) -> (__post.balances_[_to] == balances_[_to] + _value)
90    @post (msg.sender != _to) -> (__post.balances_[msg.sender] == balances_[msg.sender
        ] - _value)
91    @post (msg.sender == _to) -> (__post.balances_[_to] == balances_[_to])
92    @post (msg.sender == _to) -> (__post.balances_[msg.sender] == balances_[msg.sender
        ])
93    @post __return == true
94  */
95  function transfer(address _to, uint256 _value) public returns (bool) {
96    require(_value <= balances_[msg.sender]);
97    require(_to != address(0));
98
99    balances_[msg.sender] = balances_[msg.sender].sub(_value);
100   balances_[_to] = balances_[_to].add(_value);
101   emit Transfer(msg.sender, _to, _value);
102   return true;
103 }
104
105 /**
106  * @dev Approve the passed address to spend the specified amount of tokens on behalf
        of msg.sender.
107  * Beware that changing an allowance with this method brings the risk that someone
       may use both the old
108  * and the new allowance by unfortunate transaction ordering. One possible solution
       to mitigate this
109  * race condition is to first reduce the spender's allowance to 0 and set the
       desired value afterwards:
110  * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
111  * @param _spender The address which will spend the funds.
112  * @param _value The amount of tokens to be spent.
113  */
114 //@CTK NO_OVERFLOW
115 //@CTK NO_BUF_OVERFLOW
116 //@CTK NO_ASF
117 /*@CTK approve
118   @tag assume_completion
119   @post (__post.allowed_[msg.sender][_spender]) == (_value)
120 */
121 function approve(address _spender, uint256 _value) public returns (bool) {
122   allowed_[msg.sender][_spender] = _value;
123   emit Approval(msg.sender, _spender, _value);
124   return true;
125 }
126
127 /**
```

```
128   * @dev Transfer tokens from one address to another
129   * @param _from address The address which you want to send tokens from
130   * @param _to address The address which you want to transfer to
131   * @param _value uint256 the amount of tokens to be transferred
132   */
133  //@CTK NO_OVERFLOW
134  //@CTK NO_BUF_OVERFLOW
135  //@CTK FAIL NO_ASF
136  /*@CTK "transferFrom"
137    @tag assume_completion
138    @pre (_to) != (address(0))
139    @pre (_value) <= (balances_[_from])
140    @pre (_value) <= (allowed_[_from][msg.sender])
141    @post (_from != _to) -> (__post.balances_[_to] == (balances_[_to] + _value))
142    @post (_from != _to) -> (__post.balances_[_from] == (balances_[_from] - _value))
143    @post (_from == _to) -> (__post.balances_[_to] == balances_[_to])
144    @post (_from == _to) -> (__post.balances_[_from] == balances_[_from])
145    @post (__post.allowed_[_from][msg.sender]) == (allowed_[_from][msg.sender] -
           _value)
146    @post (__return) == (true)
147  */
148  function transferFrom(
149    address _from,
150    address _to,
151    uint256 _value
152  )
153  public
154  returns (bool)
155  {
156    require(_value <= balances_[_from]);
157    require(_value <= allowed_[_from][msg.sender]);
158    require(_to != address(0));
159
160    balances_[_from] = balances_[_from].sub(_value);
161    balances_[_to] = balances_[_to].add(_value);
162    allowed_[_from][msg.sender] = allowed_[_from][msg.sender].sub(_value);
163    emit Transfer(_from, _to, _value);
164    return true;
165  }
166
167  /**
168   * @dev Increase the amount of tokens that an owner allowed to a spender.
169   * approve should be called when allowed_[_spender] == 0. To increment
170   * allowed value is better to use this function to avoid 2 calls (and wait until
171   * the first transaction is mined)
172   * From MonolithDAO Token.sol
173   * @param _spender The address which will spend the funds.
174   * @param _addedValue The amount of tokens to increase the allowance by.
175   */
176  //@CTK NO_OVERFLOW
177  //@CTK NO_BUF_OVERFLOW
178  //@CTK FAIL NO_ASF
179  /*@CTK increaseApproval
180    @tag assume_completion
181    @pre _spender != 0x0
182    @pre _spender != msg.sender
183    @post (__post.allowed_[msg.sender][_spender]) == (allowed_[msg.sender][_spender] +
           _addedValue)
```

```
184     @post (__return) == (true)
185   */
186   function increaseApproval(
187     address _spender,
188     uint256 _addedValue
189   )
190   public
191   returns (bool)
192   {
193     allowed_[msg.sender][_spender] = (
194     allowed_[msg.sender][_spender].add(_addedValue));
195     emit Approval(msg.sender, _spender, allowed_[msg.sender][_spender]);
196     return true;
197   }
198
199   /**
200    * @dev Decrease the amount of tokens that an owner allowed to a spender.
201    * approve should be called when allowed_[_spender] == 0. To decrement
202    * allowed value is better to use this function to avoid 2 calls (and wait until
203    * the first transaction is mined)
204    * From MonolithDAO Token.sol
205    * @param _spender The address which will spend the funds.
206    * @param _subtractedValue The amount of tokens to decrease the allowance by.
207    */
208   //@CTK NO_OVERFLOW
209   //@CTK NO_BUF_OVERFLOW
210   //@CTK NO_ASF
211   /*@CTK decreaseApproval
212     @tag assume_completion
213     @pre _spender != 0x0
214     @pre _spender != msg.sender
215     @post (_subtractedValue > allowed_[msg.sender][_spender]) -> (__post.allowed_[msg.
          sender][_spender] == 0)
216     @post (_subtractedValue <= allowed_[msg.sender][_spender]) -> (__post.allowed_[msg
          .sender][_spender] == allowed_[msg.sender][_spender] - _subtractedValue)
217   */
218   function decreaseApproval(
219     address _spender,
220     uint256 _subtractedValue
221   )
222   public
223   returns (bool)
224   {
225     uint256 oldValue = allowed_[msg.sender][_spender];
226     if (_subtractedValue >= oldValue) {
227       allowed_[msg.sender][_spender] = 0;
228     } else {
229       allowed_[msg.sender][_spender] = oldValue.sub(_subtractedValue);
230     }
231     emit Approval(msg.sender, _spender, allowed_[msg.sender][_spender]);
232     return true;
233   }
234
235   /**
236    * @dev Internal function that mints an amount of the token and assigns it to
237    * an account. This encapsulates the modification of balances such that the
238    * proper events are emitted.
239    * @param _account The account that will receive the created tokens.
```

```
240   * @param _amount The amount that will be created.
241   */
242  //@CTK NO_OVERFLOW
243  //@CTK NO_BUF_OVERFLOW
244  //@CTK FAIL NO_ASF
245  /*@CTK mint
246    @tag assume_completion
247    @pre (_account != address(0))
248    @post (__post.totalSupply_) == ((totalSupply_) + (_amount))
249    @post (__post.balances_[_account]) == ((balances_[_account]) + (_amount))
250  */
251  function _mint(address _account, uint256 _amount) internal {
252    require(_account != address(0));
253    totalSupply_ = totalSupply_.add(_amount);
254    balances_[_account] = balances_[_account].add(_amount);
255    emit Transfer(address(0), _account, _amount);
256  }
257
258  /**
259   * @dev Internal function that burns an amount of the token of a given
260   * account.
261   * @param _account The account whose tokens will be burnt.
262   * @param _amount The amount that will be burnt.
263   */
264  //@CTK NO_OVERFLOW
265  //@CTK NO_BUF_OVERFLOW
266  //@CTK FAIL NO_ASF
267  /*@CTK burn
268    @tag assume_completion
269    @pre (_account != address(0))
270    @post (_amount <= balances_[_account])
271    @post (__post.totalSupply_) == ((totalSupply_) - (_amount))
272    @post (__post.balances_[_account]) == ((balances_[_account]) - (_amount))
273  */
274  function _burn(address _account, uint256 _amount) internal {
275    require(_account != address(0));
276    require(_amount <= balances_[_account]);
277
278    totalSupply_ = totalSupply_.sub(_amount);
279    balances_[_account] = balances_[_account].sub(_amount);
280    emit Transfer(_account, address(0), _amount);
281  }
282
283  /**
284   * @dev Internal function that burns an amount of the token of a given
285   * account, deducting from the sender's allowance for said account. Uses the
286   * internal _burn function.
287   * @param _account The account whose tokens will be burnt.
288   * @param _amount The amount that will be burnt.
289   */
290  //@CTK NO_OVERFLOW
291  //@CTK NO_BUF_OVERFLOW
292  //@CTK FAIL NO_ASF
293  /*@CTK burnFrom
294    @tag assume_completion
295    @post (_amount <= allowed_[_account][msg.sender])
296    @post (_amount <= balances_[_account])
297    @post (__post.allowed_[_account][msg.sender]) == ((allowed_[_account][msg.sender])
```

```
        - (_amount))
298     @post (__post.balances_[_account]) == ((balances_[_account]) - (_amount))
299   */
300   function _burnFrom(address _account, uint256 _amount) internal {
301     require(_amount <= allowed_[_account][msg.sender]);
302
303     // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
          ,
304     // this function needs to emit an event with the updated approval.
305     allowed_[_account][msg.sender] = allowed_[_account][msg.sender].sub(
306       _amount);
307     _burn(_account, _amount);
308   }
309 }
```

File SafeMath.sol

```
1  pragma solidity ^0.5;
2
3
4  /**
5   * @title SafeMath
6   * @dev Math operations with safety checks that throw on error
7   */
8  library SafeMath {
9
10   /**
11   * @dev Multiplies two numbers, throws on overflow.
12   */
13   //@CTK FAIL NO_ASF
14   /*@CTK "SafeMath mul"
15     @post ((a > 0) && (((a * b) / a) != b)) == (__reverted)
16     @post !__reverted -> __return == a * b
17     @post !__reverted == !__has_overflow
18     @post !__reverted -> !(__has_assertion_failure)
19     @post !(__has_buf_overflow)
20     */
21   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
22     if (a == 0) {
23       return 0;
24     }
25     uint256 c = a * b;
26     assert(c / a == b);
27     return c;
28   }
29
30   /**
31   * @dev Integer division of two numbers, truncating the quotient.
32   */
33   //@CTK FAIL NO_ASF
34   /*@CTK "SafeMath div"
35     @post b != 0 -> !__reverted
36     @post !__reverted -> __return == a / b
37     @post !__reverted -> !__has_overflow
38     @post !__reverted -> !(__has_assertion_failure)
39     @post !(__has_buf_overflow)
40   */
41   function div(uint256 a, uint256 b) internal pure returns (uint256) {
42     // assert(b > 0); // Solidity automatically throws when dividing by 0
```

```
43      // uint256 c = a / b;
44      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
45      return a / b;
46    }
47
48    /**
49    * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater than
          minuend).
50    */
51    //@CTK FAIL NO_ASF
52    /*@CTK "SafeMath sub"
53      @post (a < b) == __reverted
54      @post !__reverted -> __return == a - b
55      @post !__reverted -> !__has_overflow
56      @post !__reverted -> !(__has_assertion_failure)
57      @post !(__has_buf_overflow)
58    */
59    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
60      assert(b <= a);
61      return a - b;
62    }
63
64    /**
65    * @dev Adds two numbers, throws on overflow.
66    */
67    //@CTK FAIL NO_ASF
68    /*@CTK "SafeMath add"
69      @post (a + b < a || a + b < b) == __reverted
70      @post !__reverted -> __return == a + b
71      @post !__reverted -> !__has_overflow
72      @post !__reverted -> !(__has_assertion_failure)
73      @post !(__has_buf_overflow)
74    */
75    function add(uint256 a, uint256 b) internal pure returns (uint256) {
76      uint256 c = a + b;
77      assert(c >= a);
78      return c;
79    }
80  }
```