CertiK Audit Report For SponB

Request Date: 2019-06-17
Revision Date: 2019-06-18
Platform Name: Ethereum

Contents

Disclaimer

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 SponB. This audit was conducted to discover issues and vulnerabilities in the source code of SponB'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

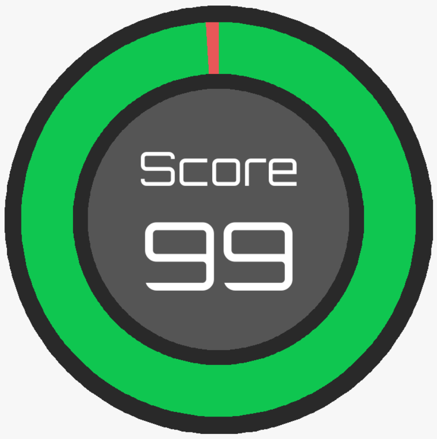
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

Testing Summary

PASS

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

Jun 18, 2019



Type of Issues

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

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Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.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

None. (Note: The violations in the formal verification result section of this report is for internal evaluation and are not indications of security issues in the user smart contracts. We recommend using `require` in place of `assert` as did in OpenZeppelin's latest `SafeMath` library.)

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **sponb.sol**
  `cef917d19ae144734eed2a5a2efe0a7216b931e28cf8717d5d6412a508fdb7d8`

Summary

CertiK was chosen by SponB to audit the design and implementation of its `SPONBToken` 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

None.

Static Analysis Results

INSECURE_COMPILER_VERSION

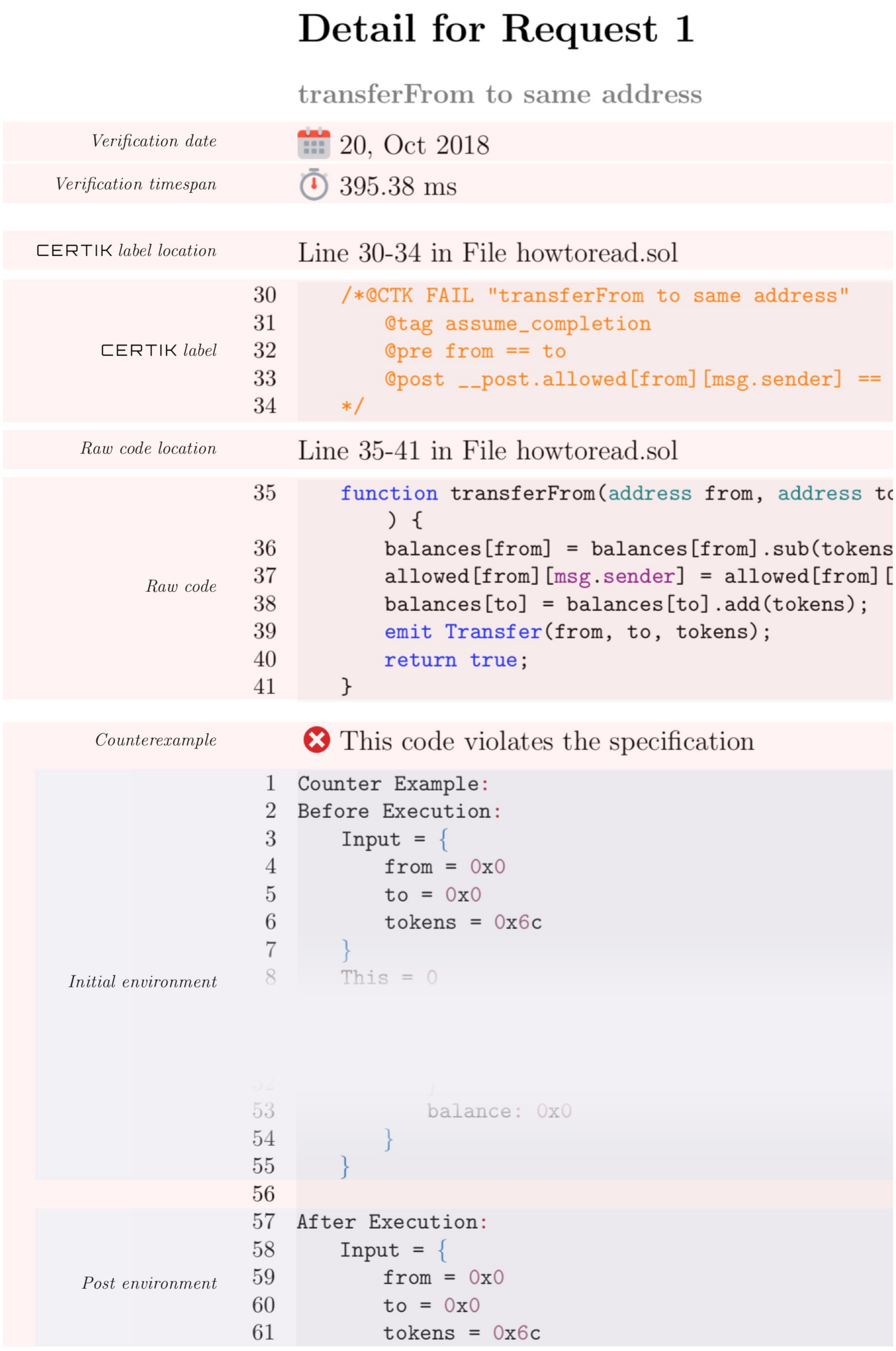
Line 1 in File sponb.sol

```
pragma solidity ^0.4.24;
```

❗ Version to compile has the following bug: 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

```
Counter Example:
Before Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
    }
    This = 0

            balance: 0x0
        }
    }

```

Post environment

```
After Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
```

Formal Verification Request 1

Method will not encounter an assertion failure.

📅 18, Jun 2019
⏱ 40.08 ms

Line 30 in File sponb.sol

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

Line 38-49 in File sponb.sol

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
    // Gas optimization: this is cheaper than asserting '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;
        }

        c = a * b;
        assert(c / a == b);
        return c;
    }
```

❌ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        a = 2
        b = 156
    }
    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
        }
        c = 0
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

```

```
Function invocation is reverted.
```

Formal Verification Request 2

SafeMath mul

📅 18, Jun 2019
⏱ 294.99 ms

Line 31-37 in File sponb.sol

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

Line 38-49 in File sponb.sol

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
    // Gas optimization: this is cheaper than asserting '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;
        }

        c = a * b;
        assert(c / a == b);
        return c;
    }
```

✓ The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.

📅 18, Jun 2019
⏱ 5.83 ms

Line 54 in File sponb.sol

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

Line 62-67 in File sponb.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;
    }
```

✗ This code violates the specification.

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

Function invocation is reverted.
```

Formal Verification Request 4

SafeMath div

18, Jun 2019
0.32 ms

Line 55-61 in File sponb.sol

```
55      /*@CTK "SafeMath div"
56        @post b != 0 -> !__reverted
57        @post !__reverted -> __return == a / b
58        @post !__reverted -> !__has_overflow
59        @post !__reverted -> !(__has_assertion_failure)
60        @post !(__has_buf_overflow)
61      */
```

Line 62-67 in File sponb.sol

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

```
    }
```

✓ The code meets the specification.

Formal Verification Request 5

Method will not encounter an assertion failure.

18, Jun 2019
11.24 ms

Line 73 in File sponb.sol

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

Line 81-84 in File sponb.sol

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

✗ This code violates the specification.

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

Function invocation is reverted.
```

Formal Verification Request 6

SafeMath sub

📅 18, Jun 2019
⏱ 0.8 ms

Line 74-80 in File sponb.sol

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

Line 81-84 in File sponb.sol

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

✅ The code meets the specification.

Formal Verification Request 7

Method will not encounter an assertion failure.

📅 18, Jun 2019
⏱ 11.92 ms

Line 90 in File sponb.sol

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

Line 98-102 in File sponb.sol

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

❌ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        a = 191
        b = 65
    }
    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
        }
        c = 0
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 8

SafeMath add

📅 18, Jun 2019
⏱ 2.7 ms

Line 91-97 in File sponb.sol

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

Line 98-102 in File sponb.sol

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

✅ The code meets the specification.

Formal Verification Request 9

If method completes, integer overflow would not happen.

📅 18, Jun 2019
⏱ 6.58 ms

Line 149 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 157-162 in File sponb.sol

```
    constructor()
    public
    {
        tokenTransfer = false;
        owner = msg.sender;
    }
```

✓ The code meets the specification.

Formal Verification Request 10

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

18, Jun 2019
0.34 ms

Line 150 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 157-162 in File sponb.sol

```
    constructor()
    public
    {
        tokenTransfer = false;
        owner = msg.sender;
    }
```

✓ The code meets the specification.

Formal Verification Request 11

Method will not encounter an assertion failure.

18, Jun 2019
0.35 ms

Line 151 in File sponb.sol

```
    //@CTK NO_ASF
```

Line 157-162 in File sponb.sol

```
    constructor()
    public
    {
        tokenTransfer = false;
        owner = msg.sender;
    }
```

✓ The code meets the specification.

Formal Verification Request 12

Lockable constructor

📅 18, Jun 2019
⏱ 0.83 ms

Line 152-156 in File sponb.sol

```
    /*@CTK "Lockable constructor"
      @tag assume_completion
      @post !__post.tokenTransfer
      @post __post.owner == msg.sender
     */
```

Line 157-162 in File sponb.sol

```
    constructor()
    public
    {
        tokenTransfer = false;
        owner = msg.sender;
    }
```

✓ The code meets the specification.

Formal Verification Request 13

If method completes, integer overflow would not happen.

📅 18, Jun 2019
⏱ 21.93 ms

Line 167 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 176-183 in File sponb.sol

```
    function setLockAddress(address target, bool status)
    external
    isOwner
    {
        require(owner != target);
        lockAddress[target] = status;
        emit Locked(target, status);
    }
```

✓ The code meets the specification.

Formal Verification Request 14

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

📅 18, Jun 2019
⏱ 0.49 ms

Line 168 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 176-183 in File sponb.sol

```
    function setLockAddress(address target, bool status)
    external
    isOwner
    {
        require(owner != target);
        lockAddress[target] = status;
        emit Locked(target, status);
    }
```

✓ The code meets the specification.

Formal Verification Request 15

Method will not encounter an assertion failure.

📅 18, Jun 2019
⏱ 0.49 ms

Line 169 in File sponb.sol

```
    //@CTK NO_ASF
```

Line 176-183 in File sponb.sol

```
    function setLockAddress(address target, bool status)
    external
    isOwner
    {
        require(owner != target);
        lockAddress[target] = status;
        emit Locked(target, status);
    }
```

✓ The code meets the specification.

Formal Verification Request 16

Lockable setLockAddress

📅 18, Jun 2019
⏱ 6.93 ms

Line 170-175 in File sponb.sol

```
    /*@CTK "Lockable setLockAddress"
      @tag assume_completion
      @post owner == msg.sender
      @post owner != target
      @post __post.lockAddress[target] == status
     */
```

Line 176-183 in File sponb.sol

```
    function setLockAddress(address target, bool status)
    external
    isOwner
    {
        require(owner != target);
        lockAddress[target] = status;
        emit Locked(target, status);
    }
```

✓ The code meets the specification.

Formal Verification Request 17

If method completes, integer overflow would not happen.

18, Jun 2019
13.94 ms

Line 188 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 196-202 in File sponb.sol

```
    function setUnlockAddress(address target, bool status)
    external
    isOwner
    {
        unlockAddress[target] = status;
        emit Unlocked(target, status);
    }
```

✓ The code meets the specification.

Formal Verification Request 18

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

18, Jun 2019
0.41 ms

Line 189 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 196-202 in File sponb.sol

```
    function setUnlockAddress(address target, bool status)
    external
    isOwner
    {
        unlockAddress[target] = status;
        emit Unlocked(target, status);
    }
```

✓ The code meets the specification.

Formal Verification Request 19

Method will not encounter an assertion failure.

📅 18, Jun 2019
⏱ 0.39 ms

Line 190 in File sponb.sol

```
    //@CTK NO_ASF
```

Line 196-202 in File sponb.sol

```
    function setUnlockAddress(address target, bool status)
    external
    isOwner
    {
        unlockAddress[target] = status;
        emit Unlocked(target, status);
    }
```

✓ The code meets the specification.

Formal Verification Request 20

Lockable setUnlockAddress

📅 18, Jun 2019
⏱ 2.72 ms

Line 191-195 in File sponb.sol

```
    /*@CTK "Lockable setUnlockAddress"
      @tag assume_completion
      @post owner == msg.sender
      @post __post.unlockAddress[target] == status
     */
```

Line 196-202 in File sponb.sol

```
    function setUnlockAddress(address target, bool status)
    external
    isOwner
    {
        unlockAddress[target] = status;
        emit Unlocked(target, status);
    }
```

✓ The code meets the specification.

Formal Verification Request 21

If method completes, integer overflow would not happen.

📅 18, Jun 2019
⏱ 18.17 ms

Line 237 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 246-253 in File sponb.sol

```
    constructor(uint256 initial_balance)
    public
    {
        require(initial_balance != 0);
        _supply = initial_balance;
        _balances[msg.sender] = initial_balance;
        emit Transfer(address(0), msg.sender, initial_balance);
    }
```

✓ The code meets the specification.

Formal Verification Request 22

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

18, Jun 2019
0.41 ms

Line 238 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 246-253 in File sponb.sol

```
    constructor(uint256 initial_balance)
    public
    {
        require(initial_balance != 0);
        _supply = initial_balance;
        _balances[msg.sender] = initial_balance;
        emit Transfer(address(0), msg.sender, initial_balance);
    }
```

✓ The code meets the specification.

Formal Verification Request 23

Method will not encounter an assertion failure.

18, Jun 2019
0.4 ms

Line 239 in File sponb.sol

```
    //@CTK NO_ASF
```

Line 246-253 in File sponb.sol

```
    constructor(uint256 initial_balance)
    public
    {
        require(initial_balance != 0);
        _supply = initial_balance;
        _balances[msg.sender] = initial_balance;
        emit Transfer(address(0), msg.sender, initial_balance);
    }
```

✓ The code meets the specification.

Formal Verification Request 24

SPONBToken

18, Jun 2019
2.66 ms

Line 240-245 in File sponb.sol

```
    /*@CTK SPONBToken
      @tag assume_completion
      @pre (initial_balance != 0)
      @post __post._supply == initial_balance
      @post __post._balances[msg.sender] == initial_balance
    */
```

Line 246-253 in File sponb.sol

```
    constructor(uint256 initial_balance)
    public
    {
        require(initial_balance != 0);
        _supply = initial_balance;
        _balances[msg.sender] = initial_balance;
        emit Transfer(address(0), msg.sender, initial_balance);
    }
```

✓ The code meets the specification.

Formal Verification Request 25

If method completes, integer overflow would not happen.

18, Jun 2019
5.36 ms

Line 255 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 262-267 in File sponb.sol

```
    function totalSupply()
    public
    view
    returns (uint256) {
        return _supply;
    }
```

✓ The code meets the specification.

Formal Verification Request 26

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

18, Jun 2019
0.36 ms

Line 256 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 262-267 in File sponb.sol

```
    function totalSupply()
    public
    view
    returns (uint256) {
        return _supply;
    }
```

✓ The code meets the specification.

Formal Verification Request 27

Method will not encounter an assertion failure.

18, Jun 2019
0.32 ms

Line 257 in File sponb.sol

```
    //@CTK NO_ASF
```

Line 262-267 in File sponb.sol

```
    function totalSupply()
    public
    view
    returns (uint256) {
        return _supply;
    }
```

✓ The code meets the specification.

Formal Verification Request 28

totalSupply

18, Jun 2019
0.34 ms

Line 258-261 in File sponb.sol

```
    /*@CTK totalSupply
      @tag assume_completion
      @post (__return) == (_supply)
    */
```

Line 262-267 in File sponb.sol

```
    function totalSupply()
    public
    view
    returns (uint256) {
        return _supply;
    }
```

✓ The code meets the specification.

Formal Verification Request 29

If method completes, integer overflow would not happen.

📅 18, Jun 2019
⏱ 5.59 ms

Line 269 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 276-281 in File sponb.sol

```
    function balanceOf(address who)
    public
    view
    returns (uint256) {
        return _balances[who];
    }
```

✓ The code meets the specification.

Formal Verification Request 30

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

📅 18, Jun 2019
⏱ 0.33 ms

Line 270 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 276-281 in File sponb.sol

```
    function balanceOf(address who)
    public
    view
    returns (uint256) {
        return _balances[who];
    }
```

✓ The code meets the specification.

Formal Verification Request 31

Method will not encounter an assertion failure.

18, Jun 2019
0.32 ms

Line 271 in File sponb.sol

```
    //@CTK NO_ASF
```

Line 276-281 in File sponb.sol

```
    function balanceOf(address who)
    public
    view
    returns (uint256) {
        return _balances[who];
    }
```

✓ The code meets the specification.

Formal Verification Request 32

balanceOf

18, Jun 2019
0.34 ms

Line 272-275 in File sponb.sol

```
    /*@CTK balanceOf
      @tag assume_completion
      @post (__return) == (_balances[who])
    */
```

Line 276-281 in File sponb.sol

```
    function balanceOf(address who)
    public
    view
    returns (uint256) {
        return _balances[who];
    }
```

✓ The code meets the specification.

Formal Verification Request 33

If method completes, integer overflow would not happen.

18, Jun 2019
133.57 ms

Line 283 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 298-310 in File sponb.sol

```
    function transfer(address to, uint256 value)
    public
    isTokenTransfer
    checkLock
    returns (bool) {
        require(to != address(0));
        require(_balances[msg.sender] >= value);

        _balances[msg.sender] = _balances[msg.sender].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(msg.sender, to, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 34

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

18, Jun 2019
31.35 ms

Line 284 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 298-310 in File sponb.sol

```
    function transfer(address to, uint256 value)
    public
    isTokenTransfer
    checkLock
    returns (bool) {
        require(to != address(0));
        require(_balances[msg.sender] >= value);

        _balances[msg.sender] = _balances[msg.sender].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(msg.sender, to, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 35

Method will not encounter an assertion failure.

18, Jun 2019
73.62 ms

Line 285 in File sponb.sol

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

Line 298-310 in File sponb.sol

```
function transfer(address to, uint256 value)
public
isTokenTransfer
checkLock
returns (bool) {
    require(to != address(0));
    require(_balances[msg.sender] >= value);

    _balances[msg.sender] = _balances[msg.sender].sub(value);
    _balances[to] = _balances[to].add(value);
    emit Transfer(msg.sender, to, value);
    return true;
}
```

❌ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        to = 8
        value = 169
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "SPONBToken",
          "balance": 0,
          "contract": {
            "name": "",
            "symbol": "",
            "decimals": 0,
            "adminMode": false,
            "_balances": [
              {
                "key": 8,
                "value": 161
              },
              {
                "key": 64,
                "value": 128
              },
              {
                "key": 1,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 169
              }
            ],
            "_approvals": [
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 169
                  }
                ]
              }
            ],
            "_supply": 0,
            "tokenTransfer": true,
            "owner": 0,
            "unlockAddress": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ],
            "lockAddress": [
              {
                "key": 32,
                "value": true
              },
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 36

transfer

📅 18, Jun 2019
⏱ 356.64 ms

Line 286-297 in File sponb.sol

```
    /*@CTK transfer
      @tag assume_completion
      @post (tokenTransfer || unlockAddress[msg.sender])
      @post !lockAddress[msg.sender]
      @pre to != address(0)
      @pre value <= _balances[msg.sender]
      @post (msg.sender != to) -> (__post._balances[to] == _balances[to] + value)
      @post (msg.sender != to) -> (__post._balances[msg.sender] == _balances[msg.
          sender] - value)
      @post (msg.sender == to) -> (__post._balances[to] == _balances[to])
      @post (msg.sender == to) -> (__post._balances[msg.sender] == _balances[msg.
          sender])
      @post __return == true
    */
```

Line 298-310 in File sponb.sol

```
    function transfer(address to, uint256 value)
    public
    isTokenTransfer
    checkLock
    returns (bool) {
        require(to != address(0));
        require(_balances[msg.sender] >= value);

        _balances[msg.sender] = _balances[msg.sender].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(msg.sender, to, value);
        return true;
    }
```

✅ The code meets the specification.

Formal Verification Request 37

If method completes, integer overflow would not happen.

📅 18, Jun 2019
⏱ 4.59 ms

Line 312 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 319-324 in File sponb.sol

```
    function allowance(address owner, address spender)
    public
    view
    returns (uint256) {
        return _approvals[owner][spender];
    }
```

✓ The code meets the specification.

Formal Verification Request 38

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

18, Jun 2019
0.3 ms

Line 313 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 319-324 in File sponb.sol

```
    function allowance(address owner, address spender)
    public
    view
    returns (uint256) {
        return _approvals[owner][spender];
    }
```

✓ The code meets the specification.

Formal Verification Request 39

Method will not encounter an assertion failure.

18, Jun 2019
0.3 ms

Line 314 in File sponb.sol

```
    //@CTK NO_ASF
```

Line 319-324 in File sponb.sol

```
    function allowance(address owner, address spender)
    public
    view
    returns (uint256) {
        return _approvals[owner][spender];
    }
```

✓ The code meets the specification.

Formal Verification Request 40

allowance

18, Jun 2019
0.3 ms

Line 315-318 in File sponb.sol

```
315    /*@CTK allowance
316      @tag assume_completion
317      @post (__return) == (_approvals[owner][spender])
318    */
```

Line 319-324 in File sponb.sol

```
319    function allowance(address owner, address spender)
320    public
321    view
322    returns (uint256) {
323        return _approvals[owner][spender];
324    }
```

✓ The code meets the specification.

Formal Verification Request 41

If method completes, integer overflow would not happen.

18, Jun 2019
157.67 ms

Line 326 in File sponb.sol

```
326    //@CTK NO_OVERFLOW
```

Line 341-354 in File sponb.sol

```
341    function transferFrom(address from, address to, uint256 value)
342    public
343    isTokenTransfer
344    checkLock
345    returns (bool success) {
346        require(!lockAddress[from]);
347        require(_balances[from] >= value);
348        require(_approvals[from][msg.sender] >= value);
349        _balances[from] = _balances[from].sub(value);
350        _balances[to] = _balances[to].add(value);
351        _approvals[from][msg.sender] = _approvals[from][msg.sender].sub(value);
352        emit Transfer(from, to, value);
353        return true;
354    }
```

✓ The code meets the specification.

Formal Verification Request 42

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

18, Jun 2019
42.81 ms

Line 327 in File sponb.sol

```
327    //@CTK NO_BUF_OVERFLOW
```

Line 341-354 in File sponb.sol

```
function transferFrom(address from, address to, uint256 value)
public
isTokenTransfer
checkLock
returns (bool success) {
    require(!lockAddress[from]);
    require(_balances[from] >= value);
    require(_approvals[from][msg.sender] >= value);
    _balances[from] = _balances[from].sub(value);
    _balances[to] = _balances[to].add(value);
    _approvals[from][msg.sender] = _approvals[from][msg.sender].sub(value);
    emit Transfer(from, to, value);
    return true;
}
```

✓ The code meets the specification.

Formal Verification Request 43

Method will not encounter an assertion failure.

18, Jun 2019
208.86 ms

Line 328 in File sponb.sol

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

Line 341-354 in File sponb.sol

```
function transferFrom(address from, address to, uint256 value)
public
isTokenTransfer
checkLock
returns (bool success) {
    require(!lockAddress[from]);
    require(_balances[from] >= value);
    require(_approvals[from][msg.sender] >= value);
    _balances[from] = _balances[from].sub(value);
    _balances[to] = _balances[to].add(value);
    _approvals[from][msg.sender] = _approvals[from][msg.sender].sub(value);
    emit Transfer(from, to, value);
    return true;
}
```

✗ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        from = 1
        to = 0
        value = 129
    }
    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
        }
        success = false
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "SPONBToken",
          "balance": 0,
          "contract": {
            "name": "",
            "symbol": "",
            "decimals": 128,
            "adminMode": false,
            "_balances": [
              {
                "key": 3,
                "value": 32
              },
              {
                "key": 0,
                "value": 159
              },
              {
                "key": 36,
                "value": 0
              },
              {
                "key": 8,
                "value": 0
              },
              {
                "key": 128,
                "value": 32
              },
              {
                "key": 4,
                "value": 0
              },
              {
                "key": 32,
                "value": 32
              },
              {
                "key": 9,
                "value": 0
              },
              {
                "key": 2,
                "value": 0
              },
              {
                "key": 129,
                "value": 0
              },
              {
                "key": 16,
                "value": 0
              },
              {
                "key": 1,
                "value": 192
              },
              {
                "key": "ALL_OTHERS",
                "value": 170
              }
            ],
            "_approvals": [
              {
                "key": 0,
                "value": [
                  {
                    "key": 128,
                    "value": 2
                  },
                  {
                    "key": 0,
                    "value": 8
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 16
                  }
                ]
              },
              {
                "key": 1,
                "value": [
                  {
                    "key": 0,
                    "value": 170
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 1
                  }
                ]
              },
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
```

```
                  "key": "ALL_OTHERS",
                  "value": 170
                }
              ]
            }
          ],
          "_supply": 4,
          "tokenTransfer": false,
          "owner": 128,
          "unlockAddress": [
            {
              "key": 32,
              "value": true
            },
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "lockAddress": [
            {
              "key": 128,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 44

transferFrom

📅 18, Jun 2019
⏱ 1025.32 ms

Line 329-340 in File sponb.sol

```
    /*@CTK "transferFrom"
      @tag assume_completion
      @pre !lockAddress[from]
      @pre (value) <= (_balances[from])
```

```
333      @pre (value) <= (_approvals[from][msg.sender])
334      @post (from != to) -> (__post._balances[to] == (_balances[to] + value))
335      @post (from != to) -> (__post._balances[from] == (_balances[from] - value))
336      @post (from == to) -> (__post._balances[to] == _balances[to])
337      @post (from == to) -> (__post._balances[from] == _balances[from])
338      @post (__post._approvals[from][msg.sender]) == (_approvals[from][msg.sender] -
             value)
339      @post (success) == (true)
340    */
```

Line 341-354 in File sponb.sol

```
341    function transferFrom(address from, address to, uint256 value)
342    public
343    isTokenTransfer
344    checkLock
345    returns (bool success) {
346        require(!lockAddress[from]);
347        require(_balances[from] >= value);
348        require(_approvals[from][msg.sender] >= value);
349        _balances[from] = _balances[from].sub(value);
350        _balances[to] = _balances[to].add(value);
351        _approvals[from][msg.sender] = _approvals[from][msg.sender].sub(value);
352        emit Transfer(from, to, value);
353        return true;
354    }
```

✓ The code meets the specification.

Formal Verification Request 45

If method completes, integer overflow would not happen.

18, Jun 2019
17.13 ms

Line 365 in File sponb.sol

```
365    //@CTK NO_OVERFLOW
```

Line 373-380 in File sponb.sol

```
373    function approve(address spender, uint256 value)
374    public
375    checkLock
376    returns (bool) {
377        _approvals[msg.sender][spender] = value;
378        emit Approval(msg.sender, spender, value);
379        return true;
380    }
```

✓ The code meets the specification.

Formal Verification Request 46

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

18, Jun 2019

⏱ 0.41 ms

Line 366 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 373-380 in File sponb.sol

```
    function approve(address spender, uint256 value)
    public
    checkLock
    returns (bool) {
        _approvals[msg.sender][spender] = value;
        emit Approval(msg.sender, spender, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 47

Method will not encounter an assertion failure.

📅 18, Jun 2019
⏱ 0.49 ms

Line 367 in File sponb.sol

```
    //@CTK NO_ASF
```

Line 373-380 in File sponb.sol

```
    function approve(address spender, uint256 value)
    public
    checkLock
    returns (bool) {
        _approvals[msg.sender][spender] = value;
        emit Approval(msg.sender, spender, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 48

approve

📅 18, Jun 2019
⏱ 2.31 ms

Line 368-372 in File sponb.sol

```
    /*@CTK approve
      @tag assume_completion
      @post !lockAddress[msg.sender]
      @post (__post._approvals[msg.sender][spender]) == (value)
    */
```

Line 373-380 in File sponb.sol

```
    function approve(address spender, uint256 value)
    public
    checkLock
    returns (bool) {
        _approvals[msg.sender][spender] = value;
        emit Approval(msg.sender, spender, value);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 49

If method completes, integer overflow would not happen.

📅 18, Jun 2019
⏱ 38.87 ms

Line 391 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 400-408 in File sponb.sol

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

✓ The code meets the specification.

Formal Verification Request 50

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

📅 18, Jun 2019
⏱ 0.85 ms

Line 392 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 400-408 in File sponb.sol

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

✓ The code meets the specification.

Formal Verification Request 51

Method will not encounter an assertion failure.

📅 18, Jun 2019
⏱ 11.18 ms

Line 393 in File sponb.sol

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

Line 400-408 in File sponb.sol

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

✗ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        _addedValue = 161
        _spender = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "SPONBToken",
          "balance": 0,
          "contract": {
            "name": "",
            "symbol": "",
            "decimals": 0,
            "adminMode": false,
            "_balances": [
              {
                "key": 0,
                "value": 16
              },
              {
                "key": 4,
                "value": 0
              },
              {
                "key": 32,
                "value": 32
              },
              {
                "key": "ALL_OTHERS",
                "value": 161
              }
            ],
            "_approvals": [
              {
                "key": 0,
                "value": [
                  {
                    "key": 0,
                    "value": 95
                  },
                  {
                    "key": 2,
                    "value": 0
                  },
                  {
                    "key": 8,
                    "value": 0
                  },
                  {
                    "key": 32,
                    "value": 0
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 161
                  }
                ]
              },
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
```

```
                  "key": "ALL_OTHERS",
                  "value": 161
                }
              ]
            }
          ],
          "_supply": 0,
          "tokenTransfer": false,
          "owner": 0,
          "unlockAddress": [
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "lockAddress": [
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 52

increaseApproval

📅 18, Jun 2019
⏱ 4.1 ms

Line 394-399 in File sponb.sol

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

Line 400-408 in File sponb.sol

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

✓ The code meets the specification.

Formal Verification Request 53

If method completes, integer overflow would not happen.

📅 18, Jun 2019
⏱ 50.46 ms

Line 419 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 429-441 in File sponb.sol

```
    function decreaseApproval(address _spender, uint256 _subtractedValue)
    public
    checkLock
    returns (bool) {
        uint256 oldValue = _approvals[msg.sender][_spender];
        if (_subtractedValue > oldValue) {
            _approvals[msg.sender][_spender] = 0;
        } else {
            _approvals[msg.sender][_spender] = oldValue.sub(_subtractedValue);
        }
        emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 54

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

📅 18, Jun 2019
⏱ 0.92 ms

Line 420 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 429-441 in File sponb.sol

```
    function decreaseApproval(address _spender, uint256 _subtractedValue)
    public
    checkLock
    returns (bool) {
        uint256 oldValue = _approvals[msg.sender][_spender];
        if (_subtractedValue > oldValue) {
            _approvals[msg.sender][_spender] = 0;
        } else {
            _approvals[msg.sender][_spender] = oldValue.sub(_subtractedValue);
        }
        emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 55

Method will not encounter an assertion failure.

18, Jun 2019
1.44 ms

Line 421 in File sponb.sol

```
    //@CTK NO_ASF
```

Line 429-441 in File sponb.sol

```
    function decreaseApproval(address _spender, uint256 _subtractedValue)
    public
    checkLock
    returns (bool) {
        uint256 oldValue = _approvals[msg.sender][_spender];
        if (_subtractedValue > oldValue) {
            _approvals[msg.sender][_spender] = 0;
        } else {
            _approvals[msg.sender][_spender] = oldValue.sub(_subtractedValue);
        }
        emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 56

decreaseApproval

18, Jun 2019
51.55 ms

Line 422-428 in File sponb.sol

```
    /*@CTK decreaseApproval
      @tag assume_completion
      @post !lockAddress[msg.sender]
      @pre _spender != msg.sender
      @post (_subtractedValue > _approvals[msg.sender][_spender]) -> (__post.
          _approvals[msg.sender][_spender] == 0)
      @post (_subtractedValue <= _approvals[msg.sender][_spender]) -> (__post.
          _approvals[msg.sender][_spender] == _approvals[msg.sender][_spender] -
          _subtractedValue)
    */
```

Line 429-441 in File sponb.sol

```
    function decreaseApproval(address _spender, uint256 _subtractedValue)
    public
    checkLock
    returns (bool) {
        uint256 oldValue = _approvals[msg.sender][_spender];
        if (_subtractedValue > oldValue) {
            _approvals[msg.sender][_spender] = 0;
        } else {
            _approvals[msg.sender][_spender] = oldValue.sub(_subtractedValue);
        }
        emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 57

If method completes, integer overflow would not happen.

📅 18, Jun 2019
⏱ 82.11 ms

Line 447 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 458-468 in File sponb.sol

```
    function burnTokens(uint256 tokensAmount)
    public
    isAdminMode
    isOwner
    {
        require(_balances[msg.sender] >= tokensAmount);

        _balances[msg.sender] = _balances[msg.sender].sub(tokensAmount);
        _supply = _supply.sub(tokensAmount);
        emit TokenBurned(msg.sender, tokensAmount);
    }
```

✓ The code meets the specification.

Formal Verification Request 58

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

📅 18, Jun 2019
⏱ 14.95 ms

Line 448 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 458-468 in File sponb.sol

```
function burnTokens(uint256 tokensAmount)
public
isAdminMode
isOwner
{
    require(_balances[msg.sender] >= tokensAmount);

    _balances[msg.sender] = _balances[msg.sender].sub(tokensAmount);
    _supply = _supply.sub(tokensAmount);
    emit TokenBurned(msg.sender, tokensAmount);
}
```

✓ The code meets the specification.

Formal Verification Request 59

Method will not encounter an assertion failure.

18, Jun 2019
42.35 ms

Line 449 in File sponb.sol

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

Line 458-468 in File sponb.sol

```
function burnTokens(uint256 tokensAmount)
public
isAdminMode
isOwner
{
    require(_balances[msg.sender] >= tokensAmount);

    _balances[msg.sender] = _balances[msg.sender].sub(tokensAmount);
    _supply = _supply.sub(tokensAmount);
    emit TokenBurned(msg.sender, tokensAmount);
}
```

✗ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        tokensAmount = 4
    }
    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": "SPONBToken",
          "balance": 0,
          "contract": {
            "name": "",
            "symbol": "",
            "decimals": 0,
            "adminMode": true,
            "_balances": [
              {
                "key": 32,
                "value": 16
              },
              {
                "key": 64,
                "value": 2
              },
              {
                "key": 1,
                "value": 0
              },
              {
                "key": 16,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 128
              }
            ],
            "_approvals": [
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 128
                  }
                ]
              }
            ],
            "_supply": 0,
            "tokenTransfer": false,
            "owner": 0,
            "unlockAddress": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ],
            "lockAddress": [
              {
                "key": 0,
                "value": true
              },
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

  Function invocation is reverted.
```

Formal Verification Request 60

burnTokens

18, Jun 2019
149.79 ms

Line 450-457 in File sponb.sol

```
    /*@CTK burnTokens
      @tag assume_completion
      @pre adminMode
      @pre owner == msg.sender
      @post (tokensAmount <= _balances[msg.sender])
      @post (__post._supply) == ((_supply) - (tokensAmount))
      @post (__post._balances[msg.sender]) == ((_balances[msg.sender]) - (tokensAmount
          ))
    */
```

Line 458-468 in File sponb.sol

```
    function burnTokens(uint256 tokensAmount)
    public
    isAdminMode
    isOwner
    {
        require(_balances[msg.sender] >= tokensAmount);

        _balances[msg.sender] = _balances[msg.sender].sub(tokensAmount);
        _supply = _supply.sub(tokensAmount);
        emit TokenBurned(msg.sender, tokensAmount);
    }
```

✓ The code meets the specification.

Formal Verification Request 61

If method completes, integer overflow would not happen.

📅 18, Jun 2019
⏱ 19.14 ms

Line 479 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 488-495 in File sponb.sol

```
    function setTokenTransfer(bool _tokenTransfer)
    external
    isAdminMode
    isOwner
    {
        tokenTransfer = _tokenTransfer;
        emit SetTokenTransfer(tokenTransfer);
    }
```

✅ The code meets the specification.

Formal Verification Request 62

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

📅 18, Jun 2019
⏱ 0.49 ms

Line 480 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 488-495 in File sponb.sol

```
    function setTokenTransfer(bool _tokenTransfer)
    external
    isAdminMode
    isOwner
    {
        tokenTransfer = _tokenTransfer;
        emit SetTokenTransfer(tokenTransfer);
    }
```

✅ The code meets the specification.

Formal Verification Request 63

Method will not encounter an assertion failure.

📅 18, Jun 2019
⏱ 0.47 ms

Line 481 in File sponb.sol

```
    //@CTK NO_ASF
```

Line 488-495 in File sponb.sol

```
    function setTokenTransfer(bool _tokenTransfer)
    external
    isAdminMode
    isOwner
    {
        tokenTransfer = _tokenTransfer;
        emit SetTokenTransfer(tokenTransfer);
    }
```

✓ The code meets the specification.

Formal Verification Request 64

setTokenTransfer

18, Jun 2019
3.24 ms

Line 482-487 in File sponb.sol

```
    /*@CTK setTokenTransfer
      @tag assume_completion
      @pre adminMode
      @pre owner == msg.sender
      @post __post.tokenTransfer == _tokenTransfer
    */
```

Line 488-495 in File sponb.sol

```
    function setTokenTransfer(bool _tokenTransfer)
    external
    isAdminMode
    isOwner
    {
        tokenTransfer = _tokenTransfer;
        emit SetTokenTransfer(tokenTransfer);
    }
```

✓ The code meets the specification.

Formal Verification Request 65

If method completes, integer overflow would not happen.

18, Jun 2019
14.09 ms

Line 497 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 505-511 in File sponb.sol

```
505     function setAdminMode(bool _adminMode)
506     public
507     isOwner
508     {
509         adminMode = _adminMode;
510         emit SetAdminMode(adminMode);
511     }
```

The code meets the specification.

Formal Verification Request 66

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

18, Jun 2019
0.37 ms

Line 498 in File sponb.sol

```
498     //@CTK NO_BUF_OVERFLOW
```

Line 505-511 in File sponb.sol

```
505     function setAdminMode(bool _adminMode)
506     public
507     isOwner
508     {
509         adminMode = _adminMode;
510         emit SetAdminMode(adminMode);
511     }
```

The code meets the specification.

Formal Verification Request 67

Method will not encounter an assertion failure.

18, Jun 2019
0.35 ms

Line 499 in File sponb.sol

```
499     //@CTK NO_ASF
```

Line 505-511 in File sponb.sol

```
505     function setAdminMode(bool _adminMode)
506     public
507     isOwner
508     {
509         adminMode = _adminMode;
510         emit SetAdminMode(adminMode);
511     }
```

The code meets the specification.

Formal Verification Request 68

setAdminMode

📅 18, Jun 2019
⏱ 2.43 ms

Line 500-504 in File sponb.sol

```
    /*@CTK setAdminMode
      @tag assume_completion
      @pre owner == msg.sender
      @post __post.adminMode == _adminMode
    */
```

Line 505-511 in File sponb.sol

```
    function setAdminMode(bool _adminMode)
    public
    isOwner
    {
        adminMode = _adminMode;
        emit SetAdminMode(adminMode);
    }
```

✅ The code meets the specification.

Formal Verification Request 69

If method completes, integer overflow would not happen.

📅 18, Jun 2019
⏱ 85.96 ms

Line 517 in File sponb.sol

```
    //@CTK NO_OVERFLOW
```

Line 528-541 in File sponb.sol

```
    function emergencyTransfer(address emergencyAddress)
    public
    isAdminMode
    isOwner
    returns (bool success) {
        require(emergencyAddress != owner);
        _balances[owner] = _balances[owner].add(_balances[emergencyAddress]);

        emit Transfer(emergencyAddress, owner, _balances[emergencyAddress]);
        emit EmergencyTransfer(emergencyAddress, owner, _balances[emergencyAddress]);

        _balances[emergencyAddress] = 0;
        return true;
    }
```

✅ The code meets the specification.

Formal Verification Request 70

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

📅 18, Jun 2019
⏱ 5.7 ms

Line 518 in File sponb.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 528-541 in File sponb.sol

```
    function emergencyTransfer(address emergencyAddress)
    public
    isAdminMode
    isOwner
    returns (bool success) {
        require(emergencyAddress != owner);
        _balances[owner] = _balances[owner].add(_balances[emergencyAddress]);

        emit Transfer(emergencyAddress, owner, _balances[emergencyAddress]);
        emit EmergencyTransfer(emergencyAddress, owner, _balances[emergencyAddress]);

        _balances[emergencyAddress] = 0;
        return true;
    }
```

✓ The code meets the specification.

Formal Verification Request 71

Method will not encounter an assertion failure.

📅 18, Jun 2019
⏱ 40.94 ms

Line 519 in File sponb.sol

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

Line 528-541 in File sponb.sol

```
    function emergencyTransfer(address emergencyAddress)
    public
    isAdminMode
    isOwner
    returns (bool success) {
        require(emergencyAddress != owner);
        _balances[owner] = _balances[owner].add(_balances[emergencyAddress]);

        emit Transfer(emergencyAddress, owner, _balances[emergencyAddress]);
        emit EmergencyTransfer(emergencyAddress, owner, _balances[emergencyAddress]);

        _balances[emergencyAddress] = 0;
        return true;
    }
```

✗ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        emergencyAddress = 64
    }
    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
        }
        success = false
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "SPONBToken",
          "balance": 0,
          "contract": {
            "name": "",
            "symbol": "",
            "decimals": 0,
            "adminMode": true,
            "_balances": [
              {
                "key": 1,
                "value": 2
              },
              {
                "key": 16,
                "value": 8
              },
              {
                "key": 0,
                "value": 192
              },
              {
                "key": 64,
                "value": 64
              },
              {
                "key": 8,
                "value": 2
              },
              {
                "key": "ALL_OTHERS",
                "value": 0
              }
            ],
            "_approvals": [
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 0
                  }
                ]
              }
            ],
            "_supply": 0,
            "tokenTransfer": false,
            "owner": 0,
            "unlockAddress": [
              {
                "key": 0,
                "value": true
              },
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ],
            "lockAddress": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 72

emergencyTransfer

18, Jun 2019

39.46 ms

Line 520-527 in File sponb.sol

```
    /*@CTK emergencyTransfer
      @tag assume_completion
      @pre adminMode
      @pre owner == msg.sender
      @pre emergencyAddress != owner
      @post __post._balances[owner] == (_balances[owner] + _balances[emergencyAddress
          ])
      @post __post._balances[emergencyAddress] == 0
    */
```

Line 528-541 in File sponb.sol

```
    function emergencyTransfer(address emergencyAddress)
    public
    isAdminMode
    isOwner
    returns (bool success) {
        require(emergencyAddress != owner);
        _balances[owner] = _balances[owner].add(_balances[emergencyAddress]);

        emit Transfer(emergencyAddress, owner, _balances[emergencyAddress]);
        emit EmergencyTransfer(emergencyAddress, owner, _balances[emergencyAddress]);

        _balances[emergencyAddress] = 0;
        return true;
    }
```

✓ The code meets the specification.

Source Code with CertiK Labels

File sponb.sol

```
pragma solidity ^0.4.24;

/**
 *Submitted for verification at Etherscan.io on 2019-05-17
*/

/**
 * @title ERC20 Interface
 */
contract ERC20 {
    function totalSupply() public view returns (uint256);
    function balanceOf(address who) public view returns (uint256);
    function transfer(address to, uint256 value) public returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);

    function allowance(address owner, address spender) public view returns (uint256);
    function transferFrom(address from, address to, uint256 value) public returns (
        bool);
    function approve(address spender, uint256 value) public returns (bool);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}
/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
library SafeMath {

    /**
    * @dev Multiplies two numbers, throws on overflow.
    */
    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath mul"
    @post ((a > 0) && (((a * b) / a) != b)) == (__reverted)
    @post !__reverted -> c == 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 c) {
    // Gas optimization: this is cheaper than asserting '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;
        }

        c = a * b;
        assert(c / a == b);
        return c;
    }

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

    /**
    * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater
        than minuend).
    */

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

    /**
    * @dev Adds two numbers, throws on overflow.
    */

    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath add"
      @post (a + b < a || a + b < b) == __reverted
      @post !__reverted -> c == 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 c) {
        c = a + b;
        assert(c >= a);
        return c;
    }
}
/**
 * @title Lockable Token
 * @author info@yggdrash.io
 */
contract Lockable {
    bool public tokenTransfer;
    address public owner;

    /**
     * @dev They can transfer even if tokenTranser flag is false.
     */
    mapping(address => bool) public unlockAddress;

    /**
     * @dev They cannot transfer even if tokenTransfer flag is true.
     */
    mapping(address => bool) public lockAddress;

    event Locked(address lockAddress, bool status);
    event Unlocked(address unlockedAddress, bool status);

    /**
     * @dev check whether can tranfer tokens or not.
     */
    modifier isTokenTransfer {
        if(!tokenTransfer) {
            require(unlockAddress[msg.sender]);
        }
        _;
    }

    /**
     * @dev check whether registered in lockAddress or not
     */
    modifier checkLock {
        require(!lockAddress[msg.sender]);
        _;
    }

    modifier isOwner
    {
        require(owner == msg.sender);
        _;
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "Lockable constructor"
      @tag assume_completion
      @post !__post.tokenTransfer
      @post __post.owner == msg.sender
     */
    constructor()
    public
    {
        tokenTransfer = false;
        owner = msg.sender;
    }

    /**
     * @dev add or remove in lockAddress(blacklist)
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "Lockable setLockAddress"
      @tag assume_completion
      @post owner == msg.sender
      @post owner != target
      @post __post.lockAddress[target] == status
     */
    function setLockAddress(address target, bool status)
    external
    isOwner
    {
        require(owner != target);
        lockAddress[target] = status;
        emit Locked(target, status);
    }

    /**
     * @dev add or remove in unlockAddress(whitelist)
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "Lockable setUnlockAddress"
      @tag assume_completion
      @post owner == msg.sender
      @post __post.unlockAddress[target] == status
     */
    function setUnlockAddress(address target, bool status)
    external
    isOwner
    {
        unlockAddress[target] = status;
        emit Unlocked(target, status);
    }
}
/**
 * @title YGGDRASH Token Contract.
 * @author info@yggdrash.io
 * @notice This contract is the updated version that fixes the unlocking bug.
 * This source code is audited by external auditors.
 */
contract SPONBToken is ERC20, Lockable {

    string public constant name = "SPONB";
    string public constant symbol = "SPO";
    uint8 public constant decimals = 18;

    /**
     * @dev If this flag is true, admin can use enableTokenTranfer(),
         emergencyTransfer().
     */
    bool public adminMode;

    using SafeMath for uint256;

    mapping(address => uint256) internal _balances;
    mapping(address => mapping(address => uint256)) internal _approvals;
    uint256 internal _supply;

    event TokenBurned(address burnAddress, uint256 amountOfTokens);
    event SetTokenTransfer(bool transfer);
    event SetAdminMode(bool adminMode);
    event EmergencyTransfer(address indexed from, address indexed to, uint256 value);

    modifier isAdminMode {
        require(adminMode);
        _;
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK SPONBToken
      @tag assume_completion
      @pre (initial_balance != 0)
      @post __post._supply == initial_balance
      @post __post._balances[msg.sender] == initial_balance
    */
    constructor(uint256 initial_balance)
    public
    {
        require(initial_balance != 0);
        _supply = initial_balance;
        _balances[msg.sender] = initial_balance;
        emit Transfer(address(0), msg.sender, initial_balance);
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK totalSupply
      @tag assume_completion
      @post (__return) == (_supply)
    */
    function totalSupply()
    public
    view
    returns (uint256) {
        return _supply;
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK balanceOf
      @tag assume_completion
      @post (__return) == (_balances[who])
    */
    function balanceOf(address who)
    public
    view
    returns (uint256) {
        return _balances[who];
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK transfer
      @tag assume_completion
      @post (tokenTransfer || unlockAddress[msg.sender])
      @post !lockAddress[msg.sender]
      @pre to != address(0)
      @pre value <= _balances[msg.sender]
      @post (msg.sender != to) -> (__post._balances[to] == _balances[to] + value)
      @post (msg.sender != to) -> (__post._balances[msg.sender] == _balances[msg.
          sender] - value)
      @post (msg.sender == to) -> (__post._balances[to] == _balances[to])
      @post (msg.sender == to) -> (__post._balances[msg.sender] == _balances[msg.
          sender])
      @post __return == true
    */
    function transfer(address to, uint256 value)
    public
    isTokenTransfer
    checkLock
    returns (bool) {
        require(to != address(0));
        require(_balances[msg.sender] >= value);

        _balances[msg.sender] = _balances[msg.sender].sub(value);
        _balances[to] = _balances[to].add(value);
        emit Transfer(msg.sender, to, value);
        return true;
    }

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

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK "transferFrom"
      @tag assume_completion
      @pre !lockAddress[from]
      @pre (value) <= (_balances[from])
      @pre (value) <= (_approvals[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._approvals[from][msg.sender]) == (_approvals[from][msg.sender] -
          value)
      @post (success) == (true)
    */
    function transferFrom(address from, address to, uint256 value)
    public
    isTokenTransfer
    checkLock
    returns (bool success) {
        require(!lockAddress[from]);
        require(_balances[from] >= value);
        require(_approvals[from][msg.sender] >= value);
        _balances[from] = _balances[from].sub(value);
        _balances[to] = _balances[to].add(value);
        _approvals[from][msg.sender] = _approvals[from][msg.sender].sub(value);
        emit Transfer(from, 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
      @tag assume_completion
      @post !lockAddress[msg.sender]
      @post (__post._approvals[msg.sender][spender]) == (value)
    */
    function approve(address spender, uint256 value)
    public
    checkLock
    returns (bool) {
        _approvals[msg.sender][spender] = value;
        emit Approval(msg.sender, spender, 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 FAIL NO_ASF
/*@CTK increaseApproval
  @tag assume_completion
  @post !lockAddress[msg.sender]
  @post (__post._approvals[msg.sender][_spender]) == (_approvals[msg.sender][
      _spender] + _addedValue)
  @post (__return) == (true)
*/
function increaseApproval(address _spender, uint256 _addedValue)
public
checkLock
returns (bool) {
    _approvals[msg.sender][_spender] = (
    _approvals[msg.sender][_spender].add(_addedValue));
    emit Approval(msg.sender, _spender, _approvals[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 NO_ASF
/*@CTK decreaseApproval
  @tag assume_completion
  @post !lockAddress[msg.sender]
  @pre _spender != msg.sender
  @post (_subtractedValue > _approvals[msg.sender][_spender]) -> (__post.
      _approvals[msg.sender][_spender] == 0)
  @post (_subtractedValue <= _approvals[msg.sender][_spender]) -> (__post.
      _approvals[msg.sender][_spender] == _approvals[msg.sender][_spender] -
      _subtractedValue)
*/
function decreaseApproval(address _spender, uint256 _subtractedValue)
public
checkLock
returns (bool) {
    uint256 oldValue = _approvals[msg.sender][_spender];
    if (_subtractedValue > oldValue) {
        _approvals[msg.sender][_spender] = 0;
    } else {
        _approvals[msg.sender][_spender] = oldValue.sub(_subtractedValue);
    }
    emit Approval(msg.sender, _spender, _approvals[msg.sender][_spender]);
    return true;
}

/**
 * @dev Burn tokens can only use by owner
 */

```

```
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK burnTokens
      @tag assume_completion
      @pre adminMode
      @pre owner == msg.sender
      @post (tokensAmount <= _balances[msg.sender])
      @post (__post._supply) == ((_supply) - (tokensAmount))
      @post (__post._balances[msg.sender]) == ((_balances[msg.sender]) - (tokensAmount
          ))
    */
    function burnTokens(uint256 tokensAmount)
    public
    isAdminMode
    isOwner
    {
        require(_balances[msg.sender] >= tokensAmount);

        _balances[msg.sender] = _balances[msg.sender].sub(tokensAmount);
        _supply = _supply.sub(tokensAmount);
        emit TokenBurned(msg.sender, tokensAmount);
    }

    /**
     * @dev Set the tokenTransfer flag.
     * If true,
     * - unregistered lockAddress can transfer()
     * - registered lockAddress can not transfer()
     * If false,
     * - registered unlockAddress & unregistered lockAddress
     * - can transfer(), unregistered unlockAddress can not transfer()
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK setTokenTransfer
      @tag assume_completion
      @pre adminMode
      @pre owner == msg.sender
      @post __post.tokenTransfer == _tokenTransfer
    */
    function setTokenTransfer(bool _tokenTransfer)
    external
    isAdminMode
    isOwner
    {
        tokenTransfer = _tokenTransfer;
        emit SetTokenTransfer(tokenTransfer);
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK setAdminMode
      @tag assume_completion
      @pre owner == msg.sender
      @post __post.adminMode == _adminMode
    */
    function setAdminMode(bool _adminMode)
    public
    isOwner
    {
        adminMode = _adminMode;
        emit SetAdminMode(adminMode);
    }

    /**
     * @dev In emergency situation,
     * admin can use emergencyTransfer() for protecting user's token.
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK emergencyTransfer
      @tag assume_completion
      @pre adminMode
      @pre owner == msg.sender
      @pre emergencyAddress != owner
      @post __post._balances[owner] == (_balances[owner] + _balances[emergencyAddress
          ])
      @post __post._balances[emergencyAddress] == 0
    */
    function emergencyTransfer(address emergencyAddress)
    public
    isAdminMode
    isOwner
    returns (bool success) {
        require(emergencyAddress != owner);
        _balances[owner] = _balances[owner].add(_balances[emergencyAddress]);

        emit Transfer(emergencyAddress, owner, _balances[emergencyAddress]);
        emit EmergencyTransfer(emergencyAddress, owner, _balances[emergencyAddress]);

        _balances[emergencyAddress] = 0;
        return true;
    }
}
```