CertiK Verification Report For Soda

Request Date: 2018-01-27
Revision Date: 2019-01-31

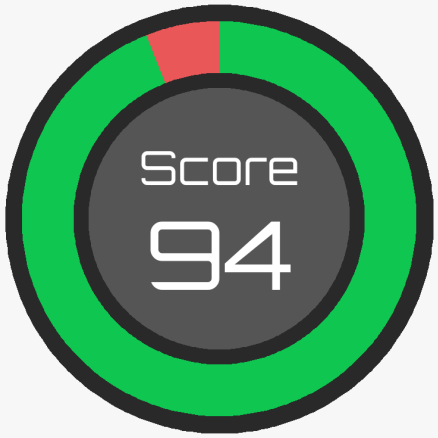
Disclaimer

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 Soda(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.

PASS

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

Jan 31, 2019



Summary

This is the report for smart contract verification service requestd by Soda. The goal of the audition is to guarantee that verified smart contracts are robust enough to avoid potentially unexpected loopholes.
The result of this report is only a reflection of the source code that was determined in this scope, and of the source code at the audit time.

Type of Issues

CertiK smart label engine applied 100% coveraged formal verification labels on the source code, and scanned the code by 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.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120
"tx.origin" for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

Owner can issue more coins after contract construction.

This is not a vulnerability. But one thing that some exchanges are concerned about. Generally if contract owner can issue more tokens, it can reduce the value of the token owner.

`TransferFrom` reduces sender's allowance even when sender = receiver

When `from` is the same as `to`, `transferFrom` reduces the sender allowance, even though there is no change in the transaction.

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.

Source Code with CertiK Labels

File sodacoin.sol

```
1  pragma solidity ^0.4.24;
2
3
4  contract SafeMath {
5      /*@CTK "SafeMath add"
6        @post (a + b < a || a + b < b) == __reverted
7        @post !__reverted -> c == a + b
8        @post !__reverted -> !__has_overflow
9      */
10     function safeAdd(uint a, uint b) public pure returns (uint c) {
11         c = a + b;
12         require(c >= a);
13     }
14     /*@CTK "SafeMath sub"
15       @post (a < b) == __reverted
16       @post !__reverted -> c == a - b
17       @post !__reverted -> !__has_overflow
18     */
19     function safeSub(uint a, uint b) public pure returns (uint c) {
20         require(b <= a);
21         c = a - b;
22     }
23     /*@CTK "SafeMath mul"
24       @post (a > 0) && (((a * b) / a) != b) -> __reverted
25       @post __reverted -> (a > 0) && (((a * b) / a) != b)
26       @post !__reverted -> c == a * b
27       @post !__reverted == !__has_overflow
28     */
29     function safeMul(uint a, uint b) public pure returns (uint c) {
30         c = a * b;
31         require(a == 0 || c / a == b);
32     }
33     /*@CTK "SafeMath div"
34       @post b != 0 -> !__reverted
35       @post !__reverted -> c == a / b
36       @post !__reverted -> !__has_overflow
37     */
38     function safeDiv(uint a, uint b) public pure returns (uint c) {
39         require(b > 0);
40         c = a / b;
41     }
42 }
43
44
45 // ----------------------------------------------------------------------------
46 // ERC Token Standard #20 Interface
47 // ----------------------------------------------------------------------------
48 contract ERC20Interface {
49     function totalSupply() public constant returns (uint);
50     function balanceOf(address tokenOwner) public constant returns (uint balance);
51     function allowance(address tokenOwner, address spender) public constant returns (
          uint remaining);
52     function transfer(address to, uint tokens) public returns (bool success);
53     function approve(address spender, uint tokens) public returns (bool success);
```

```
54      function checkRate() public constant returns (uint rate_);
55
56      event Transfer(address indexed from, address indexed to, uint tokens);
57      event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
58      event Blacklisted(address indexed target);
59    event DeleteFromBlacklist(address indexed target);
60    event RejectedPaymentToBlacklistedAddr(address indexed from, address indexed to,
          uint value);
61    event RejectedPaymentFromBlacklistedAddr(address indexed from, address indexed to,
          uint value);
62    event RejectedPaymentFromLockedAddr(address indexed from, address indexed to, uint
          value, uint lackdatetime, uint now_);
63    event RejectedPaymentMaximunFromLockedAddr(address indexed from, address indexed to,
           uint value);
64    event test1(uint rate, uint a, uint now );
65  }
66
67
68  // ----------------------------------------------------------------------------
69  // Contract function to receive approval and execute function in one call
70  // ----------------------------------------------------------------------------
71  contract ApproveAndCallFallBack {
72      function receiveApproval(address from, uint256 tokens, address token, bytes data)
            public;
73  }
74
75
76  // ----------------------------------------------------------------------------
77  // Owned contract
78  // ----------------------------------------------------------------------------
79  contract Owned {
80      address public owner;
81      address public newOwner;
82
83      event OwnershipTransferred(address indexed _from, address indexed _to);
84
85      /*@CTK Ownable
86        @post __post.owner == msg.sender
87      */
88      constructor() public {
89          owner = msg.sender;
90      }
91
92      modifier onlyOwner {
93          require(msg.sender == owner);
94          _;
95      }
96
97      /*@CTK transferOwnership
98        @tag assume_completion
99        @post owner == msg.sender
100       @post __post.newOwner == _newOwner
101     */
102     function transferOwnership(address _newOwner) public onlyOwner {
103         newOwner = _newOwner;
104     }
105     /*@CTK acceptOwnership
106       @tag assume_completion
```

```
107       @post msg.sender == newOwner
108       @post __post.owner == newOwner
109       @post __post.newOwner == address(0)
110     */
111     function acceptOwnership() public {
112         require(msg.sender == newOwner);
113         emit OwnershipTransferred(owner, newOwner);
114         owner = newOwner;
115         newOwner = address(0);
116     }
117 }
118
119
120 // ----------------------------------------------------------------------------
121 // ERC20 Token, with the addition of symbol, name and decimals and assisted
122 // token transfers
123 // ----------------------------------------------------------------------------
124 contract SodaCoin is ERC20Interface, Owned, SafeMath {
125     string public symbol;
126     string public name;
127     uint8 public decimals;
128     uint public _totalSupply;
129     uint public start;
130     address founderAddr = 0x625f7Ae05DC8c22dA56F47CaDc8c647137a6B4D9;
131     address advisorAddr = 0x45F6a7D7903D3A02bef15826eBCA44aB5eD11758;
132
133     mapping(address => uint) balances;
134     mapping(address => mapping(address => uint)) allowed;
135     mapping(address => int8) public blacklist;
136     UnlockDateModel[] public unlockdate;
137
138     struct UnlockDateModel {
139     //string date;
140     uint256 datetime;
141     uint rate;
142   }
143
144     // ------------------------------------------------------------------------
145     // Constructor
146     // ------------------------------------------------------------------------
147     constructor() public {
148         symbol = "SOC";
149         name = "SODA Coin";
150         decimals = 18;
151         _totalSupply = 2000000000000000000000000000;
152         balances[0x1E7A12b193D18027E33cd3Ff0eef2Af31cbBF9ef] =
                1400000000000000000000000000;
153         emit Transfer(address(0), 0x1E7A12b193D18027E33cd3Ff0eef2Af31cbBF9ef,
                1400000000000000000000000000); // owner wallet (70%) 1,400,000,000
154         // Founder & Team wallet (15%) 300,000,000
155         // Vesting over 2 years and 10 months (10% monthly release after 2 years)
156         balances[founderAddr] = 300000000000000000000000000;
157         emit Transfer(address(0), founderAddr, 300000000000000000000000000);
158         // Advisor & Partner wallet (15%) 300,000,000
159         // Vesting over 2 years and 10 months (10% monthly release after 2 years)
160         balances[advisorAddr] = 300000000000000000000000000;
161         emit Transfer(address(0), advisorAddr, 300000000000000000000000000);
162
```

```
163         start = now;
164         unlockdate.push(UnlockDateModel({datetime : 1610237400,rate : 10}));
165         unlockdate.push(UnlockDateModel({datetime : 1612915800,rate : 10}));
166         unlockdate.push(UnlockDateModel({datetime : 1615335000,rate : 10}));
167         unlockdate.push(UnlockDateModel({datetime : 1618013400,rate : 10}));
168         unlockdate.push(UnlockDateModel({datetime : 1620605400,rate : 10}));
169         unlockdate.push(UnlockDateModel({datetime : 1623283800,rate : 10}));
170         unlockdate.push(UnlockDateModel({datetime : 1625875800,rate : 10}));
171         unlockdate.push(UnlockDateModel({datetime : 1628554200,rate : 10}));
172         unlockdate.push(UnlockDateModel({datetime : 1631232600,rate : 10}));
173         unlockdate.push(UnlockDateModel({datetime : 1633824600,rate : 10}));
174     }
175     /*@CTK now_
176       @post __return == now
177     */
178     function now_() public constant returns (uint){
179         return now;
180     }
181 
182     // ------------------------------------------------------------------------
183     // Total supply
184     // ------------------------------------------------------------------------
185     /*@CTK totalSupply
186       @post __return == _totalSupply - balances[address(0)]
187     */
188     function totalSupply() public constant returns (uint) {
189         return _totalSupply - balances[address(0)];
190     }
191 
192     // ------------------------------------------------------------------------
193     // Get the token balance for account tokenOwner
194     // ------------------------------------------------------------------------
195     /*@CTK balanceOf
196       @post balance == balances[tokenOwner]
197     */
198     function balanceOf(address tokenOwner) public constant returns (uint balance) {
199         return balances[tokenOwner];
200     }
201 
202     function checkRate() public constant returns (uint rate_){
203         uint rate = 0;
204         for (uint i = 0; i<unlockdate.length; i++) {
205             if (unlockdate[i].datetime < now) {
206                 rate = rate + unlockdate[i].rate;
207             }
208         }
209         return rate;
210     }
211 
212     // ------------------------------------------------------------------------
213     // Transfer the balance from token owner's account to to account
214     // - Owner's account must have sufficient balance to transfer
215     // - 0 value transfers are allowed
216     // ------------------------------------------------------------------------
217 
218     function transfer(address to, uint tokens) public returns (bool success) {
219         if (msg.sender == founderAddr || msg.sender == advisorAddr){
220             if (unlockdate[0].datetime > now) {
```

```
221                 emit RejectedPaymentFromLockedAddr(msg.sender, to, tokens, unlockdate
                        [0].datetime, now);
222           return false;
223             } else {
224                 uint rate = checkRate();
225
226                 uint maximum = 300000000000000000000000000 -
                        (300000000000000000000000000 * 0.01) * rate;
227                 if (maximum > (balances[msg.sender] - tokens)){
228                     emit RejectedPaymentMaximunFromLockedAddr(msg.sender, to, tokens);
229               return false;
230                 }
231             }
232         }
233
234         if (blacklist[msg.sender] > 0) { // Accounts in the blacklist can not be
              withdrawn
235       emit RejectedPaymentFromBlacklistedAddr(msg.sender, to, tokens);
236       return false;
237     } else if (blacklist[to] > 0) { // Accounts in the blacklist can not be withdrawn
238       emit RejectedPaymentToBlacklistedAddr(msg.sender, to, tokens);
239       return false;
240     } else {
241       balances[msg.sender] = safeSub(balances[msg.sender], tokens);
242             balances[to] = safeAdd(balances[to], tokens);
243             emit Transfer(msg.sender, to, tokens);
244             return true;
245     }
246
247     }
248
249     // ------------------------------------------------------------------------
250     // Token owner can approve for spender to transferFrom(...) tokens
251     // from the token owner's account
252     // ------------------------------------------------------------------------
253     /*@CTK approve
254       @post __post.allowed[msg.sender][spender] == tokens
255     */
256     function approve(address spender, uint tokens) public returns (bool success) {
257         allowed[msg.sender][spender] = tokens;
258         emit Approval(msg.sender, spender, tokens);
259         return true;
260     }
261
262
263
264     // ------------------------------------------------------------------------
265     // Returns the amount of tokens approved by the owner that can be
266     // transferred to the spender's account
267     // ------------------------------------------------------------------------
268     /*@CTK allowance
269       @post remaining == allowed[tokenOwner][spender]
270     */
271     function allowance(address tokenOwner, address spender) public constant returns (
            uint remaining) {
272         return allowed[tokenOwner][spender];
273     }
274
```

```
275
276     // ------------------------------------------------------------------------
277     // Token owner can approve for spender to transferFrom(...) tokens
278     // from the token owner's account. The spender contract function
279     // receiveApproval(...) is then executed
280     // ------------------------------------------------------------------------
281     function approveAndCall(address spender, uint tokens, bytes data) public returns (
            bool success) {
282         allowed[msg.sender][spender] = tokens;
283         emit Approval(msg.sender, spender, tokens);
284         ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, this, data)
                ;
285         return true;
286     }
287
288
289     // ------------------------------------------------------------------------
290     // Don't accept ETH
291     // ------------------------------------------------------------------------
292     function () public payable {
293         revert();
294     }
295
296     // ------------------------------------------------------------------------
297     // Owner can transfer out any accidentally sent ERC20 tokens
298     // ------------------------------------------------------------------------
299     function transferAnyERC20Token(address tokenAddress, uint tokens) public onlyOwner
            returns (bool success) {
300         return ERC20Interface(tokenAddress).transfer(owner, tokens);
301     }
302
303     // ------------------------------------------------------------------------
304     // Owner can add an increase total supply.
305     // ------------------------------------------------------------------------
306     /*@CTK totalSupplyIncrease
307       @tag assume_completion
308       @post __post._totalSupply == _totalSupply + _supply
309       @post __post.balances[msg.sender] == balances[msg.sender] + _supply
310     */
311   function totalSupplyIncrease(uint256 _supply) public onlyOwner{
312     _totalSupply = _totalSupply + _supply;
313     balances[msg.sender] = balances[msg.sender] + _supply;
314   }
315
316   // ------------------------------------------------------------------------
317     // Owner can add blacklist the wallet address.
318     // ------------------------------------------------------------------------
319     /*@CTK blacklisting
320       @tag assume_completion
321       @post owner == msg.sender
322       @post __post.blacklist[_addr] == 1
323     */
324   function blacklisting(address _addr) public onlyOwner{
325     blacklist[_addr] = 1;
326     emit Blacklisted(_addr);
327   }
328
329
```

```
  // ---------------------------------------------------------------------------
    // Owner can delete from blacklist the wallet address.
    // ---------------------------------------------------------------------------
    /*@CTK deleteFromBlacklist
      @tag assume_completion
      @post owner == msg.sender
      @post __post.blacklist[_addr] == -1
    */
  function deleteFromBlacklist(address _addr) public onlyOwner{
    blacklist[_addr] = -1;
    emit DeleteFromBlacklist(_addr);
  }

}
```

File sodatoken.sol

```
pragma solidity ^0.4.24;


contract SafeMath {
    /*@CTK "SafeMath add"
      @post (a + b < a || a + b < b) == __reverted
      @post !__reverted -> c == a + b
      @post !__reverted -> !__has_overflow
    */
    function safeAdd(uint a, uint b) public pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    /*@CTK "SafeMath sub"
      @post (a < b) == __reverted
      @post !__reverted -> c == a - b
      @post !__reverted -> !__has_overflow
    */
    function safeSub(uint a, uint b) public pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
    /*@CTK "SafeMath mul"
      @post (a > 0) && (((a * b) / a) != b) -> __reverted
      @post __reverted -> (a > 0) && (((a * b) / a) != b)
      @post !__reverted -> c == a * b
      @post !__reverted == !__has_overflow
    */
    function safeMul(uint a, uint b) public pure returns (uint c) {
        c = a * b;
        require(a == 0 || c / a == b);
    }
    /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> c == a / b
      @post !__reverted -> !__has_overflow
    */
    function safeDiv(uint a, uint b) public pure returns (uint c) {
        require(b > 0);
        c = a / b;
    }
}
```

```
43
44
45  // ----------------------------------------------------------------------------
46  // ERC Token Standard #20 Interface
47  // ----------------------------------------------------------------------------
48  contract ERC20Interface {
49      function totalSupply() public constant returns (uint);
50      function balanceOf(address tokenOwner) public constant returns (uint balance);
51      function allowance(address tokenOwner, address spender) public constant returns (
            uint remaining);
52      function transfer(address to, uint tokens) public returns (bool success);
53      function approve(address spender, uint tokens) public returns (bool success);
54      function transferFrom(address from, address to, uint tokens) public returns (bool
            success);
55
56      event Transfer(address indexed from, address indexed to, uint tokens);
57      event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
58  }
59
60
61  // ----------------------------------------------------------------------------
62  // Contract function to receive approval and execute function in one call
63  // ----------------------------------------------------------------------------
64  contract ApproveAndCallFallBack {
65      function receiveApproval(address from, uint256 tokens, address token, bytes data)
            public;
66  }
67
68
69  // ----------------------------------------------------------------------------
70  // Owned contract
71  // ----------------------------------------------------------------------------
72  contract Owned {
73      address public owner;
74      address public newOwner;
75
76      event OwnershipTransferred(address indexed _from, address indexed _to);
77
78      /*@CTK Ownable
79        @post __post.owner == msg.sender
80      */
81      constructor() public {
82          owner = msg.sender;
83      }
84
85      modifier onlyOwner {
86          require(msg.sender == owner);
87          _;
88      }
89
90      /*@CTK transferOwnership
91        @tag assume_completion
92        @post owner == msg.sender
93        @post __post.newOwner == _newOwner
94      */
95      function transferOwnership(address _newOwner) public onlyOwner {
96          newOwner = _newOwner;
97      }
```

```
98      /*@CTK acceptOwnership
99        @tag assume_completion
100       @post msg.sender == newOwner
101       @post __post.owner == newOwner
102       @post __post.newOwner == address(0)
103     */
104     function acceptOwnership() public {
105         require(msg.sender == newOwner);
106         emit OwnershipTransferred(owner, newOwner);
107         owner = newOwner;
108         newOwner = address(0);
109     }
110 }
111
112
113 // ----------------------------------------------------------------------------
114 // ERC20 Token, with the addition of symbol, name and decimals and assisted
115 // token transfers
116 // ----------------------------------------------------------------------------
117 contract SodaToken is ERC20Interface, Owned, SafeMath {
118     string public symbol;
119     string public name;
120     uint8 public decimals;
121     uint public _totalSupply;
122
123     mapping(address => uint) balances;
124     mapping(address => mapping(address => uint)) allowed;
125
126
127     // ------------------------------------------------------------------------
128     // Constructor
129     // ------------------------------------------------------------------------
130     constructor() public {
131         symbol = "SOT";
132         name = "SODA Token";
133         decimals = 18;
134         _totalSupply = 200000000000000000000000000;
135         balances[0xC713b7c600Bb0e70c2d4b466b923Cab1E45e7c76] = _totalSupply;
136         emit Transfer(address(0), 0xC713b7c600Bb0e70c2d4b466b923Cab1E45e7c76,
              _totalSupply);
137     }
138
139
140     // ------------------------------------------------------------------------
141     // Total supply
142     // ------------------------------------------------------------------------
143     /*@CTK totalSupply
144       @post __return == _totalSupply - balances[address(0)]
145     */
146     function totalSupply() public constant returns (uint) {
147         return _totalSupply - balances[address(0)];
148     }
149
150
151     // ------------------------------------------------------------------------
152     // Get the token balance for account tokenOwner
153     // ------------------------------------------------------------------------
154     /*@CTK balanceOf
```

```
155       @post balance == balances[tokenOwner]
156     */
157     function balanceOf(address tokenOwner) public constant returns (uint balance) {
158         return balances[tokenOwner];
159     }
160
161
162     // ------------------------------------------------------------------------
163     // Transfer the balance from token owner's account to to account
164     // - Owner's account must have sufficient balance to transfer
165     // - 0 value transfers are allowed
166     // ------------------------------------------------------------------------
167     /*@CTK transfer
168       @tag assume_completion
169       @pre to != msg.sender
170       @post __post.balances[msg.sender] == balances[msg.sender] - tokens
171       @post __post.balances[to] == balances[to] + tokens
172     */
173     function transfer(address to, uint tokens) public returns (bool success) {
174         balances[msg.sender] = safeSub(balances[msg.sender], tokens);
175         balances[to] = safeAdd(balances[to], tokens);
176         emit Transfer(msg.sender, to, tokens);
177         return true;
178     }
179
180
181     // ------------------------------------------------------------------------
182     // Token owner can approve for spender to transferFrom(...) tokens
183     // from the token owner's account
184     // ------------------------------------------------------------------------
185     /*@CTK approve
186       @post __post.allowed[msg.sender][spender] == tokens
187     */
188     function approve(address spender, uint tokens) public returns (bool success) {
189         allowed[msg.sender][spender] = tokens;
190         emit Approval(msg.sender, spender, tokens);
191         return true;
192     }
193
194
195     // ------------------------------------------------------------------------
196     // Transfer tokens from the from account to the to account
197     //
198     // The calling account must already have sufficient tokens approve(...)-d
199     // for spending from the from account and
200     // - From account must have sufficient balance to transfer
201     // - Spender must have sufficient allowance to transfer
202     // - 0 value transfers are allowed
203     // ------------------------------------------------------------------------
204     /*@CTK transferFrom
205       @tag assume_completion
206       @pre to != from
207       @post __post.balances[from] == balances[from] - tokens
208       @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
209       @post __post.balances[to] == balances[to] + tokens
210     */
211     function transferFrom(address from, address to, uint tokens) public returns (bool
          success) {
```

```
212         balances[from] = safeSub(balances[from], tokens);
213         allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
214         balances[to] = safeAdd(balances[to], tokens);
215         emit Transfer(from, to, tokens);
216         return true;
217     }
218
219
220     // ------------------------------------------------------------------------
221     // Returns the amount of tokens approved by the owner that can be
222     // transferred to the spender's account
223     // ------------------------------------------------------------------------
224     /*@CTK allowance
225       @post remaining == allowed[tokenOwner][spender]
226     */
227     function allowance(address tokenOwner, address spender) public constant returns (
            uint remaining) {
228         return allowed[tokenOwner][spender];
229     }
230
231
232     // ------------------------------------------------------------------------
233     // Token owner can approve for spender to transferFrom(...) tokens
234     // from the token owner's account. The spender contract function
235     // receiveApproval(...) is then executed
236     // ------------------------------------------------------------------------
237     function approveAndCall(address spender, uint tokens, bytes data) public returns (
            bool success) {
238         allowed[msg.sender][spender] = tokens;
239         emit Approval(msg.sender, spender, tokens);
240         ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, this, data)
                ;
241         return true;
242     }
243
244
245     // ------------------------------------------------------------------------
246     // Don't accept ETH
247     // ------------------------------------------------------------------------
248     function () public payable {
249         revert();
250     }
251
252
253     // ------------------------------------------------------------------------
254     // Owner can transfer out any accidentally sent ERC20 tokens
255     // ------------------------------------------------------------------------
256     function transferAnyERC20Token(address tokenAddress, uint tokens) public onlyOwner
            returns (bool success) {
257         return ERC20Interface(tokenAddress).transfer(owner, tokens);
258     }
259     // Increase issuance.
260     /*@CTK totalSupplyIncrease
261       @tag assume_completion
262       @post __post._totalSupply == _totalSupply + _supply
263       @post __post.balances[msg.sender] == balances[msg.sender] + _supply
264     */
265   function totalSupplyIncrease(uint256 _supply) public onlyOwner{
```

```
266     _totalSupply = _totalSupply + _supply;
267     balances[msg.sender] = balances[msg.sender] + _supply;
268   }
269 }
```

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
```

Static Analysis Request

INSECURE_COMPILER_VERSION

Line 1 in File sodacoin.sol

```
1 pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

TIMESTAMP_DEPENDENCY

Line 179 in File sodacoin.sol

```
179         return now;
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 205 in File sodacoin.sol

```
205             if (unlockdate[i].datetime < now) {
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 220 in File sodacoin.sol

```
220             if (unlockdate[0].datetime > now) {
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 221 in File sodacoin.sol

```
221                 emit RejectedPaymentFromLockedAddr(msg.sender, to, tokens, unlockdate
                        [0].datetime, now);
```

"now" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 1 in File sodatoken.sol

```
1 pragma solidity ^0.4.24;
```

Only these compiler versions are safe to compile your code: 0.4.25

Formal Verification Request 1

SafeMath add

31, Jan 2019
18.33 ms

Line 5-9 in File sodacoin.sol

```
5   /*@CTK "SafeMath add"
6     @post (a + b < a || a + b < b) == __reverted
7     @post !__reverted -> c == a + b
8     @post !__reverted -> !__has_overflow
9   */
```

Line 10-13 in File sodacoin.sol

```
10  function safeAdd(uint a, uint b) public pure returns (uint c) {
11      c = a + b;
12      require(c >= a);
13  }
```

The code meets the specification

Formal Verification Request 2

SafeMath sub

31, Jan 2019
13.41 ms

Line 14-18 in File sodacoin.sol

```
14  /*@CTK "SafeMath sub"
15    @post (a < b) == __reverted
16    @post !__reverted -> c == a - b
17    @post !__reverted -> !__has_overflow
18  */
```

Line 19-22 in File sodacoin.sol

```
19  function safeSub(uint a, uint b) public pure returns (uint c) {
20      require(b <= a);
21      c = a - b;
22  }
```

The code meets the specification

Formal Verification Request 3

SafeMath mul

31, Jan 2019
139.14 ms

Line 23-28 in File sodacoin.sol

```
23    /*@CTK "SafeMath mul"
24      @post (a > 0) && (((a * b) / a) != b) -> __reverted
25      @post __reverted -> (a > 0) && (((a * b) / a) != b)
26      @post !__reverted -> c == a * b
27      @post !__reverted == !__has_overflow
28    */
```

Line 29-32 in File sodacoin.sol

```
29    function safeMul(uint a, uint b) public pure returns (uint c) {
30        c = a * b;
31        require(a == 0 || c / a == b);
32    }
```

The code meets the specification

Formal Verification Request 4

SafeMath div

31, Jan 2019
15.23 ms

Line 33-37 in File sodacoin.sol

```
33    /*@CTK "SafeMath div"
34      @post b != 0 -> !__reverted
35      @post !__reverted -> c == a / b
36      @post !__reverted -> !__has_overflow
37    */
```

Line 38-41 in File sodacoin.sol

```
38    function safeDiv(uint a, uint b) public pure returns (uint c) {
39        require(b > 0);
40        c = a / b;
41    }
```

The code meets the specification

Formal Verification Request 5

Ownable

31, Jan 2019
5.62 ms

Line 85-87 in File sodacoin.sol

```
85    /*@CTK Ownable
86      @post __post.owner == msg.sender
87    */
```

Line 88-90 in File sodacoin.sol

```
88      constructor() public {
89          owner = msg.sender;
90      }
```

The code meets the specification

Formal Verification Request 6

transferOwnership

31, Jan 2019
14.8 ms

Line 97-101 in File sodacoin.sol

```
97      /*@CTK transferOwnership
98        @tag assume_completion
99        @post owner == msg.sender
100       @post __post.newOwner == _newOwner
101     */
```

Line 102-104 in File sodacoin.sol

```
102     function transferOwnership(address _newOwner) public onlyOwner {
103         newOwner = _newOwner;
104     }
```

The code meets the specification

Formal Verification Request 7

acceptOwnership

31, Jan 2019
17.69 ms

Line 105-110 in File sodacoin.sol

```
105     /*@CTK acceptOwnership
106       @tag assume_completion
107       @post msg.sender == newOwner
108       @post __post.owner == newOwner
109       @post __post.newOwner == address(0)
110     */
```

Line 111-116 in File sodacoin.sol

```
111     function acceptOwnership() public {
112         require(msg.sender == newOwner);
113         emit OwnershipTransferred(owner, newOwner);
114         owner = newOwner;
115         newOwner = address(0);
116     }
```

The code meets the specification

Formal Verification Request 8

now_

31, Jan 2019
5.33 ms

Line 175-177 in File sodacoin.sol

```
175    /*@CTK now_
176      @post __return == now
177    */
```

Line 178-180 in File sodacoin.sol

```
178    function now_() public constant returns (uint){
179        return now;
180    }
```

The code meets the specification

Formal Verification Request 9

totalSupply

31, Jan 2019
7.57 ms

Line 185-187 in File sodacoin.sol

```
185    /*@CTK totalSupply
186      @post __return == _totalSupply - balances[address(0)]
187    */
```

Line 188-190 in File sodacoin.sol

```
188    function totalSupply() public constant returns (uint) {
189        return _totalSupply - balances[address(0)];
190    }
```

The code meets the specification

Formal Verification Request 10

balanceOf

31, Jan 2019
5.66 ms

Line 195-197 in File sodacoin.sol

```
195    /*@CTK balanceOf
196      @post balance == balances[tokenOwner]
197    */
```

Line 198-200 in File sodacoin.sol

```
198     function balanceOf(address tokenOwner) public constant returns (uint balance) {
199         return balances[tokenOwner];
200     }
```

The code meets the specification

Formal Verification Request 11

approve

31, Jan 2019
14.6 ms

Line 253-255 in File sodacoin.sol

```
253     /*@CTK approve
254       @post __post.allowed[msg.sender][spender] == tokens
255     */
```

Line 256-260 in File sodacoin.sol

```
256     function approve(address spender, uint tokens) public returns (bool success) {
257         allowed[msg.sender][spender] = tokens;
258         emit Approval(msg.sender, spender, tokens);
259         return true;
260     }
```

The code meets the specification

Formal Verification Request 12

allowance

31, Jan 2019
5.78 ms

Line 268-270 in File sodacoin.sol

```
268     /*@CTK allowance
269       @post remaining == allowed[tokenOwner][spender]
270     */
```

Line 271-273 in File sodacoin.sol

```
271     function allowance(address tokenOwner, address spender) public constant returns (
            uint remaining) {
272         return allowed[tokenOwner][spender];
273     }
```

The code meets the specification

Formal Verification Request 13

totalSupplyIncrease

31, Jan 2019
66.67 ms

Line 306-310 in File sodacoin.sol

```
306   /*@CTK totalSupplyIncrease
307     @tag assume_completion
308     @post __post._totalSupply == _totalSupply + _supply
309     @post __post.balances[msg.sender] == balances[msg.sender] + _supply
310   */
```

Line 311-314 in File sodacoin.sol

```
311  function totalSupplyIncrease(uint256 _supply) public onlyOwner{
312    _totalSupply = _totalSupply + _supply;
313    balances[msg.sender] = balances[msg.sender] + _supply;
314  }
```

The code meets the specification

Formal Verification Request 14

blacklisting

31, Jan 2019
20.57 ms

Line 319-323 in File sodacoin.sol

```
319   /*@CTK blacklisting
320     @tag assume_completion
321     @post owner == msg.sender
322     @post __post.blacklist[_addr] == 1
323   */
```

Line 324-327 in File sodacoin.sol

```
324  function blacklisting(address _addr) public onlyOwner{
325    blacklist[_addr] = 1;
326    emit Blacklisted(_addr);
327  }
```

The code meets the specification

Formal Verification Request 15

deleteFromBlacklist

31, Jan 2019
19.98 ms

Line 333-337 in File sodacoin.sol

```
333   /*@CTK deleteFromBlacklist
334     @tag assume_completion
335     @post owner == msg.sender
336     @post __post.blacklist[_addr] == -1
337   */
```

Line 338-341 in File sodacoin.sol

```
338  function deleteFromBlacklist(address _addr) public onlyOwner{
339    blacklist[_addr] = -1;
340    emit DeleteFromBlacklist(_addr);
341  }
```

The code meets the specification

Formal Verification Request 16

SafeMath add

31, Jan 2019
18.33 ms

Line 5-9 in File sodatoken.sol

```
5   /*@CTK "SafeMath add"
6     @post (a + b < a || a + b < b) == __reverted
7     @post !__reverted -> c == a + b
8     @post !__reverted -> !__has_overflow
9   */
```

Line 10-13 in File sodatoken.sol

```
10  function safeAdd(uint a, uint b) public pure returns (uint c) {
11      c = a + b;
12      require(c >= a);
13  }
```

The code meets the specification

Formal Verification Request 17

SafeMath sub

31, Jan 2019
13.41 ms

Line 14-18 in File sodatoken.sol

```
14  /*@CTK "SafeMath sub"
15    @post (a < b) == __reverted
16    @post !__reverted -> c == a - b
17    @post !__reverted -> !__has_overflow
18  */
```

Line 19-22 in File sodatoken.sol

```
19    function safeSub(uint a, uint b) public pure returns (uint c) {
20        require(b <= a);
21        c = a - b;
22    }
```

The code meets the specification

Formal Verification Request 18

SafeMath mul

31, Jan 2019
139.14 ms

Line 23-28 in File sodatoken.sol

```
23    /*@CTK "SafeMath mul"
24      @post (a > 0) && (((a * b) / a) != b) -> __reverted
25      @post __reverted -> (a > 0) && (((a * b) / a) != b)
26      @post !__reverted -> c == a * b
27      @post !__reverted == !__has_overflow
28    */
```

Line 29-32 in File sodatoken.sol

```
29    function safeMul(uint a, uint b) public pure returns (uint c) {
30        c = a * b;
31        require(a == 0 || c / a == b);
32    }
```

The code meets the specification

Formal Verification Request 19

SafeMath div

31, Jan 2019
15.23 ms

Line 33-37 in File sodatoken.sol

```
33    /*@CTK "SafeMath div"
34      @post b != 0 -> !__reverted
35      @post !__reverted -> c == a / b
36      @post !__reverted -> !__has_overflow
37    */
```

Line 38-41 in File sodatoken.sol

```
38    function safeDiv(uint a, uint b) public pure returns (uint c) {
39        require(b > 0);
40        c = a / b;
41    }
```

The code meets the specification

Formal Verification Request 20

Ownable

31, Jan 2019
5.62 ms

Line 78-80 in File sodatoken.sol

```
78    /*@CTK Ownable
79      @post __post.owner == msg.sender
80    */
```

Line 81-83 in File sodatoken.sol

```
81    constructor() public {
82        owner = msg.sender;
83    }
```

The code meets the specification

Formal Verification Request 21

transferOwnership

31, Jan 2019
14.8 ms

Line 90-94 in File sodatoken.sol

```
90    /*@CTK transferOwnership
91      @tag assume_completion
92      @post owner == msg.sender
93      @post __post.newOwner == _newOwner
94    */
```

Line 95-97 in File sodatoken.sol

```
95    function transferOwnership(address _newOwner) public onlyOwner {
96        newOwner = _newOwner;
97    }
```

The code meets the specification

Formal Verification Request 22

acceptOwnership

31, Jan 2019
17.69 ms

Line 98-103 in File sodatoken.sol

```
98    /*@CTK acceptOwnership
99      @tag assume_completion
100     @post msg.sender == newOwner
101     @post __post.owner == newOwner
102     @post __post.newOwner == address(0)
103   */
```

Line 104-109 in File sodatoken.sol

```
104   function acceptOwnership() public {
105       require(msg.sender == newOwner);
106       emit OwnershipTransferred(owner, newOwner);
107       owner = newOwner;
108       newOwner = address(0);
109   }
```

The code meets the specification

Formal Verification Request 23

totalSupply

31, Jan 2019
7.96 ms

Line 143-145 in File sodatoken.sol

```
143   /*@CTK totalSupply
144     @post __return == _totalSupply - balances[address(0)]
145   */
```

Line 146-148 in File sodatoken.sol

```
146   function totalSupply() public constant returns (uint) {
147       return _totalSupply - balances[address(0)];
148   }
```

The code meets the specification

Formal Verification Request 24

balanceOf

31, Jan 2019
5.9 ms

Line 154-156 in File sodatoken.sol

```
154   /*@CTK balanceOf
155     @post balance == balances[tokenOwner]
156   */
```

Line 157-159 in File sodatoken.sol

```
157   function balanceOf(address tokenOwner) public constant returns (uint balance) {
158       return balances[tokenOwner];
159   }
```

The code meets the specification

Formal Verification Request 25

transfer

31, Jan 2019
149.24 ms

Line 167-172 in File sodatoken.sol

```
167     /*@CTK transfer
168       @tag assume_completion
169       @pre to != msg.sender
170       @post __post.balances[msg.sender] == balances[msg.sender] - tokens
171       @post __post.balances[to] == balances[to] + tokens
172     */
```

Line 173-178 in File sodatoken.sol

```
173     function transfer(address to, uint tokens) public returns (bool success) {
174         balances[msg.sender] = safeSub(balances[msg.sender], tokens);
175         balances[to] = safeAdd(balances[to], tokens);
176         emit Transfer(msg.sender, to, tokens);
177         return true;
178     }
```

The code meets the specification

Formal Verification Request 26

approve

31, Jan 2019
12.2 ms

Line 185-187 in File sodatoken.sol

```
185     /*@CTK approve
186       @post __post.allowed[msg.sender][spender] == tokens
187     */
```

Line 188-192 in File sodatoken.sol

```
188     function approve(address spender, uint tokens) public returns (bool success) {
189         allowed[msg.sender][spender] = tokens;
190         emit Approval(msg.sender, spender, tokens);
191         return true;
192     }
```

The code meets the specification

Formal Verification Request 27

transferFrom

31, Jan 2019
199.45 ms

Line 204-210 in File sodatoken.sol

```
204   /*@CTK transferFrom
205     @tag assume_completion
206     @pre to != from
207     @post __post.balances[from] == balances[from] - tokens
208     @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - tokens
209     @post __post.balances[to] == balances[to] + tokens
210   */
```

Line 211-217 in File sodatoken.sol

```
211   function transferFrom(address from, address to, uint tokens) public returns (bool
          success) {
212       balances[from] = safeSub(balances[from], tokens);
213       allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
214       balances[to] = safeAdd(balances[to], tokens);
215       emit Transfer(from, to, tokens);
216       return true;
217   }
```

The code meets the specification

Formal Verification Request 28

allowance

31, Jan 2019
6.11 ms

Line 224-226 in File sodatoken.sol

```
224   /*@CTK allowance
225     @post remaining == allowed[tokenOwner][spender]
226   */
```

Line 227-229 in File sodatoken.sol

```
227   function allowance(address tokenOwner, address spender) public constant returns (
          uint remaining) {
228       return allowed[tokenOwner][spender];
229   }
```

The code meets the specification

Formal Verification Request 29

totalSupplyIncrease

31, Jan 2019
43.54 ms

Line 260-264 in File sodatoken.sol

```
260   /*@CTK totalSupplyIncrease
261     @tag assume_completion
262     @post __post._totalSupply == _totalSupply + _supply
263     @post __post.balances[msg.sender] == balances[msg.sender] + _supply
264   */
```

Line 265-268 in File sodatoken.sol

```
265  function totalSupplyIncrease(uint256 _supply) public onlyOwner{
266    _totalSupply = _totalSupply + _supply;
267    balances[msg.sender] = balances[msg.sender] + _supply;
268  }
```

The code meets the specification