CertiK Verification Report For GatheringChain

Request Date: 2019-01-20
Revision Date: 2019-01-22
For Exchange: OKEx

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 GatheringChain(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 CertiKs prior written consent.
CertiK consents to the Customer sharing a copy of this Report to OKEx (the "Exchange") on the understanding that such sharing will be for information purpose only without any representation, warranty or liability whatsoever on CertiK. By receiving a copy of this Report, the Exchange understands and agrees that none of CertiK, its affiliates, and its and its affiliates respective directors, officers, employees, agents and representatives shall have any liability to the Exchange or any other person whether direct or indirect, in contract, tort or otherwise, regardless of the nature of the allegation or claim, arising from or in connection with this Report, its delivery to the Exchange or any other person or its use thereof. CertiK is not, by means of permitting the Exchange to obtain this Report, rendering any advice or services to, or acting as a fiduciary or agent of, or in any other capacity with respect to the Exchange. This Report is not a substitute for such advice or services, nor should they be used as a basis for any decision or action that may affect the Exchange and its business. The Services may have been, and may in the future be, changed or modified by mutual agreement of CertiK and the Company subsequent to the date hereof. The Exchange is not a third-party beneficiary of the Agreement and the Exchange has no right thereunder or with respect to the Services.

PASS

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

Jan 22, 2019

Summary

This is the report for smart contract verification service requestd by GatheringChain. The goal of the audition is to guarantee that verified smart contracts are robust enough to avoid potentially unexpected loopholes.
Decimal of this contract is 18.

Source Code with CertiK Labels

File gac.sol

```
1  pragma solidity ^0.4.25;
2
3  interface tokenRecipient { function receiveApproval(address _from, uint256 _value,
       address _token, bytes _extraData) external; }
4
5  contract GAC {
6      // Public variables of the token
7      string public name;
8      string public symbol;
9      uint8 public decimals = 18;
10     // 18 decimals is the strongly suggested default, avoid changing it
11     uint256 public totalSupply;
12
13     // This creates an array with all balances
14     mapping (address => uint256) public balanceOf;
15     mapping (address => mapping (address => uint256)) public allowance;
16
17     // This generates a public event on the blockchain that will notify clients
18     event Transfer(address indexed from, address indexed to, uint256 value);
19
20     // This generates a public event on the blockchain that will notify clients
21     event Approval(address indexed _owner, address indexed _spender, uint256 _value);
22
23
24
25     /**
26      * Constructor function
27      *
28      * Initializes contract with initial supply tokens to the creator of the contract
29      */
30     //@CTK NO_BUF_OVERFLOW
31     //@CTK NO_ASF
32     constructor(
33         uint256 initialSupply,
34         string tokenName,
35         string tokenSymbol
36     ) public {
37         totalSupply = initialSupply * 10 ** uint256(decimals); // Update total supply
               with the decimal amount
38         balanceOf[msg.sender] = totalSupply;           // Give the creator all initial
               tokens
39         name = tokenName;                              // Set the name for display
               purposes
40         symbol = tokenSymbol;                          // Set the symbol for display
               purposes
41     }
42
43     /**
44      * Internal transfer, only can be called by this contract
45      */
46     function _transfer(address _from, address _to, uint _value) internal {
47         // Prevent transfer to 0x0 address. Use burn() instead
48         require(_to != 0x0);
49         // Check if the sender has enough
```

```
50          require(balanceOf[_from] >= _value);
51          // Check for overflows
52          require(balanceOf[_to] + _value >= balanceOf[_to]);
53          // Save this for an assertion in the future
54          uint previousBalances = balanceOf[_from] + balanceOf[_to];
55          // Subtract from the sender
56          balanceOf[_from] -= _value;
57          // Add the same to the recipient
58          balanceOf[_to] += _value;
59          emit Transfer(_from, _to, _value);
60          // Asserts are used to use static analysis to find bugs in your code. They
                should never fail
61          assert(balanceOf[_from] + balanceOf[_to] == previousBalances);
62      }
63
64      /**
65       * Transfer tokens
66       *
67       * Send `_value` tokens to `_to` from your account
68       *
69       * @param _to The address of the recipient
70       * @param _value the amount to send
71       */
72      //@CTK NO_BUF_OVERFLOW
73      //@CTK NO_ASF
74      /*@CTK "transfer_same"
75        @pre (__reverted) == (false)
76        @pre (balanceOf[msg.sender]) >= (_value)
77        @pre (balanceOf[_to] + _value) >= (balanceOf[_to])
78        @pre (balanceOf[_to] + balanceOf[msg.sender]) >= (balanceOf[_to])
79        @pre (_to) == (msg.sender)
80        @post (__post.balanceOf[_to]) == (balanceOf[_to])
81        @post (success) == (true)
82        @post (!__has_overflow)
83      */
84      /*@CTK "transfer"
85        @pre (__reverted) == (false)
86        @pre (balanceOf[msg.sender]) >= (_value)
87        @pre (balanceOf[_to] + _value) >= (balanceOf[_to])
88        @pre (balanceOf[_to] + balanceOf[msg.sender]) >= (balanceOf[_to])
89        @pre (_to) != (msg.sender)
90        @post (__post.balanceOf[_to]) == ((balanceOf[_to]) + (_value))
91        @post (__post.balanceOf[msg.sender]) == ((balanceOf[msg.sender]) - (_value))
92        @post (success) == (true)
93        @post (!__has_overflow)
94      */
95      function transfer(address _to, uint256 _value) public returns (bool success) {
96          _transfer(msg.sender, _to, _value);
97          return true;
98      }
99
100     /**
101      * Transfer tokens from other address
102      *
103      * Send `_value` tokens to `_to` on behalf of `_from`
104      *
105      * @param _from The address of the sender
106      * @param _to The address of the recipient
```

```
107      * @param _value the amount to send
108      */
109     //@CTK NO_BUF_OVERFLOW
110     //@CTK NO_ASF
111     /*@CTK "transferFrom_same"
112       @pre (__reverted) == (false)
113       @pre (balanceOf[_from]) >= (_value)
114       @pre (balanceOf[_to] + _value) >= (balanceOf[_to])
115       @pre (balanceOf[_to] + balanceOf[_from]) >= (balanceOf[_to])
116       @pre (_value <= allowance[_from][msg.sender])
117       @pre (_to) == (_from)
118       @post (__post.balanceOf[_to]) == (balanceOf[_to])
119       @post (success) == (true)
120       @post (!__has_overflow)
121     */
122     /*@CTK "transferFrom"
123       @pre (__reverted) == (false)
124       @pre (balanceOf[_from]) >= (_value)
125       @pre (balanceOf[_to] + _value) >= (balanceOf[_to])
126       @pre (balanceOf[_to] + balanceOf[_from]) >= (balanceOf[_to])
127       @pre (_value <= allowance[_from][msg.sender])
128       @pre (_to) != (_from)
129       @post (__post.balanceOf[_to]) == ((balanceOf[_to]) + (_value))
130       @post (__post.balanceOf[_from]) == ((balanceOf[_from]) - (_value))
131       @post (success) == (true)
132       @post (!__has_overflow)
133     */
134     function transferFrom(address _from, address _to, uint256 _value) public returns (
          bool success) {
135         require(_value <= allowance[_from][msg.sender]); // Check allowance
136         allowance[_from][msg.sender] -= _value;
137         _transfer(_from, _to, _value);
138         return true;
139     }
140 
141     /**
142      * Set allowance for other address
143      *
144      * Allows `_spender` to spend no more than `_value` tokens on your behalf
145      *
146      * @param _spender The address authorized to spend
147      * @param _value the max amount they can spend
148      */
149     //@CTK NO_OVERFLOW
150     //@CTK NO_BUF_OVERFLOW
151     //@CTK NO_ASF
152     /*@CTK approve
153       @tag assume_completion
154       @post __post.allowance[msg.sender][_spender] == _value
155       @post success == true
156      */
157     function approve(address _spender, uint256 _value) public
158         returns (bool success) {
159         allowance[msg.sender][_spender] = _value;
160         emit Approval(msg.sender, _spender, _value);
161         return true;
162     }
163 
```

```
164     /**
165      * Set allowance for other address and notify
166      *
167      * Allows `_spender` to spend no more than `_value` tokens on your behalf, and
             then ping the contract about it
168      *
169      * @param _spender The address authorized to spend
170      * @param _value the max amount they can spend
171      * @param _extraData some extra information to send to the approved contract
172      */
173     function approveAndCall(address _spender, uint256 _value, bytes _extraData)
174         public
175         returns (bool success) {
176         tokenRecipient spender = tokenRecipient(_spender);
177         if (approve(_spender, _value)) {
178             spender.receiveApproval(msg.sender, _value, this, _extraData);
179             return true;
180         }
181     }
182
183
184 }
```