CertiK Verification Report For IoTeX

Request Date: 2018-02-27
Revision Date: 2019-03-04

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 IoTeX(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.*

Mar 04, 2019

Summary

This audit report summarises the smart contract verification service requested by IoTeX. The goal of this security audit is to guarantee that the audited smart contracts are robust enough to avoid any potential security 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 time of the audit.

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Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0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

No issue found.

Comment

The overall code quality is high, and truly shows the engineering skills and efforts applied on this staking project. The design of the project is derived from erc900, and with many improvements and adjustments to fit the need of IoTeX voting and staking. The complexity is above average given the facts that more complex data structures (bucket class with double linked list), many for-loops, some assembly logics (split address string

to less gas usage) were introduced, however we believe those topics are well handled in the source code, such as the data structure implementations were correctly coded and most of the for-loops are getter functions which won't change the states. The documents describing the project by IoTeX released to public were also reviewed by CertiK team to ensure the specifications match the implementation, and the parts differed were either updated in the source code or edited in the articles at a fast timing.

There are a few critical parts (bonus allocation and auto-staking mechanism) not involved in this audit report, as they are handled in another layer outside the scope of the smart contract. We look forward that IoTeX team will expose the handlings of such to its supporters and involved community for better transparency and decentralization.

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 IOTX.sol

```
pragma solidity ^0.4.23;

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

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

  /*@CTK SafeMath_mul
    @tag spec
    @post __reverted == __has_assertion_failure
    @post __has_assertion_failure == __has_overflow
    @post __reverted == false -> c == a * b
    @post msg == msg__post
   */
  /* CertiK Smart Labelling, for more details visit: https://certik.org */
  function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
    if (a == 0) {
      return 0;
    }
    c = a * b;
    assert(c / a == b);
    return c;
  }

  /**
  * @dev Integer division of two numbers, truncating the quotient.
  */
  /*@CTK SafeMath_div
    @tag spec
    @pre b != 0
    @post __reverted == __has_assertion_failure
    @post __has_overflow == true -> __has_assertion_failure == true
    @post __reverted == false -> __return == a / b
    @post msg == msg__post
   */
  /* CertiK Smart Labelling, for more details visit: https://certik.org */
  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 SafeMath_sub
    @tag spec
```

```
54      @post __reverted == __has_assertion_failure
55      @post __has_overflow == true -> __has_assertion_failure == true
56      @post __reverted == false -> __return == a - b
57      @post msg == msg__post
58     */
59    /* CertiK Smart Labelling, for more details visit: https://certik.org */
60    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
61      assert(b <= a);
62      return a - b;
63    }
64
65    /**
66    * @dev Adds two numbers, throws on overflow.
67    */
68    /*@CTK SafeMath_add
69      @tag spec
70      @post __reverted == __has_assertion_failure
71      @post __has_assertion_failure == __has_overflow
72      @post __reverted == false -> c == a + b
73      @post msg == msg__post
74     */
75    /* CertiK Smart Labelling, for more details visit: https://certik.org */
76    function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
77      c = a + b;
78      assert(c >= a);
79      return c;
80    }
81  }
82
83  /**
84   * @title Ownable
85   * @dev The Ownable contract has an owner address, and provides basic authorization
        control
86   * functions, this simplifies the implementation of "user permissions".
87   */
88  contract Ownable {
89    address public owner;
90
91
92    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
93
94
95    /**
96     * @dev The Ownable constructor sets the original `owner` of the contract to the
          sender
97     * account.
98     */
99    /*@CTK owner_set_on_success
100     @pre __reverted == false -> __post.owner == owner
101    */
102   /* CertiK Smart Labelling, for more details visit: https://certik.org */
103   function Ownable() public {
104     owner = msg.sender;
105   }
106
107   /**
108    * @dev Throws if called by any account other than the owner.
109    */
```

```
110   modifier onlyOwner() {
111     require(msg.sender == owner);
112     _;
113   }
114
115   /**
116    * @dev Allows the current owner to transfer control of the contract to a newOwner.
117    * @param newOwner The address to transfer ownership to.
118    */
119   /*@CTK transferOwnership
120     @post __reverted == false -> (msg.sender == owner -> __post.owner == newOwner)
121     @post (owner != msg.sender) -> (__reverted == true)
122     @post (newOwner == address(0)) -> (__reverted == true)
123    */
124   /* CertiK Smart Labelling, for more details visit: https://certik.org */
125   function transferOwnership(address newOwner) public onlyOwner {
126     require(newOwner != address(0));
127     emit OwnershipTransferred(owner, newOwner);
128     owner = newOwner;
129   }
130
131 }
132
133 /**
134  * @title Pausable
135  * @dev Base contract which allows children to implement an emergency stop mechanism.
136  */
137 contract Pausable is Ownable {
138   event Pause();
139   event Unpause();
140
141   bool public paused = false;
142
143
144   /**
145    * @dev Modifier to make a function callable only when the contract is not paused.
146    */
147   modifier whenNotPaused() {
148     require(!paused);
149     _;
150   }
151
152   /**
153    * @dev Modifier to make a function callable only when the contract is paused.
154    */
155   modifier whenPaused() {
156     require(paused);
157     _;
158   }
159
160   /**
161    * @dev called by the owner to pause, triggers stopped state
162    */
163   function pause() onlyOwner whenNotPaused public {
164     paused = true;
165     emit Pause();
166   }
167
```

```
168   /**
169    * @dev called by the owner to unpause, returns to normal state
170    */
171   function unpause() onlyOwner whenPaused public {
172     paused = false;
173     emit Unpause();
174   }
175 }
176
177
178 /**
179  * @title ERC20Basic
180  * @dev Simpler version of ERC20 interface
181  * @dev see https://github.com/ethereum/EIPs/issues/179
182  */
183 contract ERC20Basic {
184   function totalSupply() public view returns (uint256);
185   function balanceOf(address who) public view returns (uint256);
186   function transfer(address to, uint256 value) public returns (bool);
187   event Transfer(address indexed from, address indexed to, uint256 value);
188 }
189
190 /**
191  * @title ERC20 interface
192  * @dev see https://github.com/ethereum/EIPs/issues/20
193  */
194 contract ERC20 is ERC20Basic {
195   function allowance(address owner, address spender) public view returns (uint256);
196   function transferFrom(address from, address to, uint256 value) public returns (bool)
        ;
197   function approve(address spender, uint256 value) public returns (bool);
198   event Approval(address indexed owner, address indexed spender, uint256 value);
199 }
200
201 /**
202  * @title Basic token
203  * @dev Basic version of StandardToken, with no allowances.
204  */
205 contract BasicToken is ERC20Basic {
206   using SafeMath for uint256;
207
208   mapping(address => uint256) balances;
209
210   uint256 totalSupply_;
211
212   /**
213   * @dev total number of tokens in existence
214   */
215   function totalSupply() public view returns (uint256) {
216     return totalSupply_;
217   }
218
219   /**
220   * @dev transfer token for a specified address
221   * @param _to The address to transfer to.
222   * @param _value The amount to be transferred.
223   */
224   /*@CTK transfer_success
```

```
225     @tag assume_completion
226     @post _to != address(0)
227     @post balances[msg.sender] >= _value
228    */
229   /*CTK transfer_conditions
230     @tag assume_completion
231     @pre _to != msg.sender
232     @post __post.balances[_to] == balances[_to] + _value
233     @post __post.balances[msg.sender] == balances[msg.sender] - _value
234    */
235   /* CertiK Smart Labelling, for more details visit: https://certik.org */
236   function transfer(address _to, uint256 _value) public returns (bool) {
237     require(_to != address(0));
238     require(_value <= balances[msg.sender]);
239
240     balances[msg.sender] = balances[msg.sender].sub(_value);
241     balances[_to] = balances[_to].add(_value);
242     emit Transfer(msg.sender, _to, _value);
243     return true;
244   }
245
246   /**
247   * @dev Gets the balance of the specified address.
248   * @param _owner The address to query the the balance of.
249   * @return An uint256 representing the amount owned by the passed address.
250   */
251   /*@CTK balanceOf
252     @post __reverted == false
253     @post __return == balances[_owner]
254    */
255   /* CertiK Smart Labelling, for more details visit: https://certik.org */
256   function balanceOf(address _owner) public view returns (uint256) {
257     return balances[_owner];
258   }
259
260 }
261
262 /**
263  * @title Standard ERC20 token
264  *
265  * @dev Implementation of the basic standard token.
266  * @dev https://github.com/ethereum/EIPs/issues/20
267  * @dev Based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master
       /smart_contract/FirstBloodToken.sol
268  */
269 contract StandardToken is ERC20, BasicToken {
270
271   mapping (address => mapping (address => uint256)) internal allowed;
272
273
274   /**
275    * @dev Transfer tokens from one address to another
276    * @param _from address The address which you want to send tokens from
277    * @param _to address The address which you want to transfer to
278    * @param _value uint256 the amount of tokens to be transferred
279    */
280   /*CTK transferFrom
281     @tag assume_completion
```

```
282     @pre _from != _to
283     @post __post.balances[_to] == balances[_to] + _value
284     @post __post.balances[_from] == balances[_from] - _value
285    */
286   /* CertiK Smart Labelling, for more details visit: https://certik.org */
287   function transferFrom(address _from, address _to, uint256 _value) public returns (
          bool) {
288     require(_to != address(0));
289     require(_value <= balances[_from]);
290     require(_value <= allowed[_from][msg.sender]);
291
292     balances[_from] = balances[_from].sub(_value);
293     balances[_to] = balances[_to].add(_value);
294     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
295     emit Transfer(_from, _to, _value);
296     return true;
297   }
298
299   /**
300    * @dev Approve the passed address to spend the specified amount of tokens on behalf
           of msg.sender.
301    *
302    * Beware that changing an allowance with this method brings the risk that someone
          may use both the old
303    * and the new allowance by unfortunate transaction ordering. One possible solution
          to mitigate this
304    * race condition is to first reduce the spender's allowance to 0 and set the
          desired value afterwards:
305    * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
306    * @param _spender The address which will spend the funds.
307    * @param _value The amount of tokens to be spent.
308    */
309   /*@CTK approve_success
310     @post _value == 0 -> __reverted == false
311     @post allowed[msg.sender][_spender] == 0 -> __reverted == false
312    */
313   /*@CTK approve
314     @tag assume_completion
315     @post __post.allowed[msg.sender][_spender] == _value
316    */
317   /* CertiK Smart Labelling, for more details visit: https://certik.org */
318   function approve(address _spender, uint256 _value) public returns (bool) {
319     allowed[msg.sender][_spender] = _value;
320     emit Approval(msg.sender, _spender, _value);
321     return true;
322   }
323
324   /**
325    * @dev Function to check the amount of tokens that an owner allowed to a spender.
326    * @param _owner address The address which owns the funds.
327    * @param _spender address The address which will spend the funds.
328    * @return A uint256 specifying the amount of tokens still available for the spender
          .
329    */
330   function allowance(address _owner, address _spender) public view returns (uint256) {
331     return allowed[_owner][_spender];
332   }
333
```

```
334  /**
335   * @dev Increase the amount of tokens that an owner allowed to a spender.
336   *
337   * approve should be called when allowed[_spender] == 0. To increment
338   * allowed value is better to use this function to avoid 2 calls (and wait until
339   * the first transaction is mined)
340   * From MonolithDAO Token.sol
341   * @param _spender The address which will spend the funds.
342   * @param _addedValue The amount of tokens to increase the allowance by.
343   */
344  /*@CTK CtkIncreaseApprovalEffect
345    @tag assume_completion
346    @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
         _addedValue
347    @post __has_overflow == false
348   */
349  /* CertiK Smart Labelling, for more details visit: https://certik.org */
350  function increaseApproval(address _spender, uint _addedValue) public returns (bool)
       {
351    allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
352    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
353    return true;
354  }
355
356  /**
357   * @dev Decrease the amount of tokens that an owner allowed to a spender.
358   *
359   * approve should be called when allowed[_spender] == 0. To decrement
360   * allowed value is better to use this function to avoid 2 calls (and wait until
361   * the first transaction is mined)
362   * From MonolithDAO Token.sol
363   * @param _spender The address which will spend the funds.
364   * @param _subtractedValue The amount of tokens to decrease the allowance by.
365   */
366  /*@CTK CtkDecreaseApprovalEffect_1
367    @pre allowed[msg.sender][_spender] >= _subtractedValue
368    @tag assume_completion
369    @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
         _subtractedValue
370    @post __has_overflow == false
371   */
372   /*@CTK CtkDecreaseApprovalEffect_2
373    @pre allowed[msg.sender][_spender] < _subtractedValue
374    @tag assume_completion
375    @post __post.allowed[msg.sender][_spender] == 0
376    @post __has_overflow == false
377   */
378  /* CertiK Smart Labelling, for more details visit: https://certik.org */
379  function decreaseApproval(address _spender, uint _subtractedValue) public returns (
       bool) {
380    uint oldValue = allowed[msg.sender][_spender];
381    if (_subtractedValue > oldValue) {
382      allowed[msg.sender][_spender] = 0;
383    } else {
384      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
385    }
386    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
387    return true;
```

```
388   }
389
390 }
391
392 contract IoTeXNetwork is StandardToken, Pausable {
393     string public constant name = "IoTeX Network";
394     string public constant symbol = "IOTX";
395     uint8 public constant decimals = 18;
396
397     modifier validDestination(address to) {
398         require(to != address(0x0));
399         require(to != address(this) );
400         _;
401     }
402
403     function IoTeXNetwork(uint tokenTotalAmount) {
404         totalSupply_ = tokenTotalAmount;
405         balances[msg.sender] = tokenTotalAmount;
406         emit Transfer(address(0x0), msg.sender, tokenTotalAmount);
407     }
408
409     /*CTK CtkTransferNoEffect
410       @post (_to == address(0)) \/ (paused == true) -> __reverted == true
411      */
412     /*CTK CtkTransferEffect
413       @pre __reverted == false
414       @pre balances[msg.sender] >= _value
415       @pre paused == false
416       @pre __return == true
417       @pre msg.sender != _to
418       @post __post.balances[_to] == balances[_to] + _value
419       @post __has_overflow == false
420      */
421     /* CertiK Smart Labelling, for more details visit: https://certik.org */
422     function transfer(address _to, uint _value) whenNotPaused
423         validDestination(_to)
424         returns (bool) {
425         return super.transfer(_to, _value);
426     }
427
428     /*@CTK CtkTransferFromNoEffect
429       @post (_to == address(0)) \/ (paused == true) -> __reverted == true
430      */
431     /*CTK CtkTransferFromEffect
432       @tag assume_completion
433       @pre _from != _to
434       @post __post.balances[_to] == balances[_to] + _value
435       @post __post.balances[_from] == balances[_from] - _value
436       @post __has_overflow == false
437      */
438     /* CertiK Smart Labelling, for more details visit: https://certik.org */
439     function transferFrom(address _from, address _to, uint _value) whenNotPaused
440         validDestination(_to)
441         returns (bool) {
442         return super.transferFrom(_from, _to, _value);
443     }
444
445     /*@CTK CtkApproveNoEffect
```

```
446       @post (paused == true) -> __post == this
447      */
448     /*@CTK CtkApprove
449       @tag assume_completion
450       @post __post.allowed[msg.sender][_spender] == _value
451      */
452     /* CertiK Smart Labelling, for more details visit: https://certik.org */
453     function approve(address _spender, uint256 _value) public whenNotPaused
454       returns (bool) {
455       return super.approve(_spender, _value);
456     }
457
458     /*@CTK CtkIncreaseApprovalNoEffect
459       @post (paused == true) -> __reverted == true
460      */
461     /*@CTK CtkIncreaseApprovalEffect
462       @pre paused == false
463       @tag assume_completion
464       @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
              _addedValue
465       @post __has_overflow == false
466      */
467     /* CertiK Smart Labelling, for more details visit: https://certik.org */
468     function increaseApproval(address _spender, uint _addedValue) public whenNotPaused
469       returns (bool success) {
470       return super.increaseApproval(_spender, _addedValue);
471     }
472
473     /*@CTK CtkDecreaseApprovalNoEffect
474       @post (paused == true) -> __reverted == true
475      */
476     /*@CTK CtkDecreaseApprovalEffect
477       @pre allowed[msg.sender][_spender] >= _subtractedValue
478       @tag assume_completion
479       @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
              _subtractedValue
480       @post __has_overflow == false
481      */
482     /* CertiK Smart Labelling, for more details visit: https://certik.org */
483     function decreaseApproval(address _spender, uint _subtractedValue) public
            whenNotPaused
484       returns (bool success) {
485       return super.decreaseApproval(_spender, _subtractedValue);
486     }
487 }
```

File NameRegistration.sol

```
1  pragma solidity ^0.4.24;
2
3  import "./library/ERC20.sol";
4  import "./library/Ownable.sol";
5  import "./library/SafeMath.sol";
6  import "./library/Pausable.sol";
7
8  /**
9   * @title Register delegates with name and operator and reward addresses.
10  * @author IoTeX Team
11  */
```

```
12 contract NameRegistration is Pausable {
13     using SafeMath for uint256;
14
15     event Registered(uint256 idx, bytes12 name, address addr, string ioOperatorAddr,
           string ioRewardAddr, bytes data);
16
17     uint256 public nameRegistrationFee = 100 * 10 ** 18; // IOTX
18     address public feeCollector = msg.sender; // address to receive fee
19
20     ERC20 public token;          // token address this contract is used for
21     struct Candidate {
22         bytes12 name;            // candidate name for voter to vote in smart contract
23         address addr;            // candidate address on eth
24         string ioOperatorAddr;   // operator Addr on IoTeX blockchain
25         string ioRewardAddr;     // reward Addr on IoTeX blockchain
26         uint256 weight;           // weight for robot bp
27     }
28     mapping(uint256 => Candidate) public candidates; // array of all candidates
29     uint256 public candidateCount;                   // total count of candidates
30     mapping(bytes12 => uint256) public nameToIdx;  // reserve mapping: name to
           candidate index.
31     mapping(address => uint256) public addrToIdx;  // reserve mapping: address to
           candidate index.
32     mapping(bytes32 => mapping(bytes32 => uint256)) public ioAddrToIdx; // reserve
           mapping: Addr to index to check if a Addr is used.
33
34     /**
35      * @dev Constructor function
36      * @param _tokenAddr address The address of the token contract used for staking
37      */
38     /*@CTK NameRegistration
39       @post __post.candidateCount == 1
40      */
41     constructor(address _tokenAddr) public {
42         token = ERC20(_tokenAddr);
43         candidateCount = 1; // 0 is null, reserved.
44     }
45
46     /**
47      * @dev Modifier that checks io address
48      * @param _ioAddr io address
49      */
50     modifier checkIoAddr(string _ioAddr) {
51         require(bytes(_ioAddr).length >= 40 && bytes(_ioAddr).length <= 64, "io address
               is not valid");
52         _;
53     }
54
55     /**
56      * @dev Modifier that checks candidate name
57      * @param _name candidate name
58      */
59     modifier checkName(bytes12 _name) {
60         for (uint i = 0; i < 12; i++) {
61             byte c = _name[i];
62             require((c >= 0x61 && c <= 0x7a) || (c >= 0x30 && c <= 0x39) || c == 0x0, "
                   invalid candiadate name.");
63         }
```

```
64          _;
65      }
66
67      /**
68       * @dev get first two bytes32 from a given io address
69       * @param _ioAddr io address
70       * @return (bytes32, bytes32)
71       *  p1, p2 are first two parts of the Addr.
72       */
73      //CTK: Should this be the following instead?
74      // mload(add(_ioAddr, 0x00))
75      // mload(add(_ioAddr, 0x20))
76      // mload(p) <=> mem[p, p+0x20]
77      function split(string _ioAddr) internal view returns (bytes32 p1, bytes32 p2) {
78          assembly {
79              p1 := mload(add(_ioAddr,0x20))
80              p2 := mload(add(_ioAddr,0x40))
81          }
82          return (p1, p2);
83      }
84
85      /**
86       * @dev Get all candidates for a range of indexes.
87       * @param _startIndex uint256 the starting index. NOTE: index 0 is reserved.
88       * @param _limit uint256 the number of non zero buckets to fetch after the start
             index.
89       * @return (bytes12, address[], bytes32[], bytes32[])
90       *  names, addresses, ioOperatorAddr, ioRewardAddr arrays of returning data
91       */
92      function getAllCandidates(uint256 _startIndex, uint256 _limit) external view
93              returns(bytes12[] names, address[] addresses, bytes32[] ioOperatorAddr,
                    bytes32[] ioRewardAddr, uint256[] weights) {
94          //CTK: require _startIndex > 0 since index 0 is reserved?
95          require (_startIndex < candidateCount && _limit < 500, "index or limit not
                valid.");
96          uint256 limit = _limit;
97          if (_startIndex + _limit > candidateCount) {
98              limit = candidateCount - _startIndex;
99          }
100         names = new bytes12[](limit);
101         addresses = new address[](limit);
102         ioOperatorAddr = new bytes32[](limit*2);
103         ioRewardAddr = new bytes32[](limit*2);
104         weights = new uint256[](limit);
105         for (uint256 i = 0; i < limit; i++) {
106             names[i] = candidates[_startIndex+i].name;
107             addresses[i] = candidates[_startIndex+i].addr;
108             (ioOperatorAddr[i*2], ioOperatorAddr[i*2+1]) = split(candidates[_startIndex
                    +i].ioOperatorAddr);
109             (ioRewardAddr[i*2], ioRewardAddr[i*2+1]) = split(candidates[_startIndex+i].
                    ioRewardAddr);
110             weights[i] = candidates[_startIndex+i].weight;
111         }
112         return (names, addresses, ioOperatorAddr, ioRewardAddr, weights);
113     }
114
115     function getIdxByAddr(string _ioAddr) internal view returns(uint256) {
116         //CTK: (bytes32 a, bytes32 b)=split(_ioAddr);
```

```
117         var (a, b) = split(_ioAddr);
118         return ioAddrToIdx[a][b];
119     }
120
121     function setAddrIdx(string _ioAddr, uint256 idx) internal {
122         //CTK: (bytes32 a, bytes32 b)=split(_ioAddr);
123         var (a, b) = split(_ioAddr);
124         ioAddrToIdx[a][b] = idx;
125     }
126
127     /*@CTK setFeeCollector
128       @tag assume_completion
129       @post owner == msg.sender
130       @post __post.feeCollector == _addr
131      */
132     function setFeeCollector(address _addr) external onlyOwner {
133         feeCollector = _addr;
134     }
135
136     /*@CTK setNameRegistrationFee
137       @tag assume_completion
138       @post owner == msg.sender
139       @post __post.nameRegistrationFee == _fee
140      */
141     function setNameRegistrationFee(uint256 _fee) external onlyOwner {
142         nameRegistrationFee = _fee;
143     }
144
145     /*@CTK setWeight
146       @tag assume_completion
147       @post owner == msg.sender
148       @post nameToIdx[_name] > 0
149       @post __post.candidates[nameToIdx[_name]].weight == _weight
150      */
151     function setWeight(bytes12 _name, uint256 _weight) external onlyOwner {
152         uint256 idx = nameToIdx[_name];
153         require(idx > 0, "name not registered.");
154         candidates[idx].weight = _weight;
155     }
156
157     /*@CTK setNameAddress
158       @tag assume_completion
159       @post owner == msg.sender
160       @post addrToIdx[_addr] == 0
161       @post candidates[nameToIdx[_name]].addr != _addr
162       @post __post.candidates[nameToIdx[_name]].addr == _addr
163       @post __post.addrToIdx[_addr] == nameToIdx[_name]
164      */
165     //CTK: extremly slow
166     function setNameAddress(bytes12 _name, address _addr) external onlyOwner {
167         require(addrToIdx[_addr] == 0, "new addr should not have name");
168         uint256 idx = nameToIdx[_name]; // find the candidate
169         address oldAddr = candidates[idx].addr;
170         require(oldAddr != _addr, "new address is expected");
171         candidates[idx].addr = _addr;
172         addrToIdx[_addr] = idx;
173         delete addrToIdx[oldAddr];
174     }
```

```
175
176     /**
177      * @notice Register a name as a candidate and provide io operator and reward
             addresses.
178      * @notice MUST trigger Registered event
179      * @param _name string The name of the candidate
180      * @param _ioOperatorAddr string operator's address on IoTeX
181      * @param _ioRewardAddr string reward address on IoTeX
182      * @param _data bytes optional data to include in the emitted event
183      */
184     /*CTK register
185       @tag assume_completion
186       @pre addrToIdx[msg.sender] <= 0
187       @post nameToIdx[_name] == 0
188      */
189     //CTK: Very slow and need to delete checkName modifier
190     function register(bytes12 _name, string _ioOperatorAddr, string _ioRewardAddr,
            bytes _data) external whenNotPaused
191         checkIoAddr(_ioOperatorAddr) checkIoAddr(_ioRewardAddr) checkName(_name) {
192         uint256 idx;
193         if (addrToIdx[msg.sender] > 0) { // has entry, updating
194             idx = addrToIdx[msg.sender];
195             if (nameToIdx[_name] == 0) { // not taken
196                 delete nameToIdx[candidates[idx].name]; // delete old index.
197                 candidates[idx].name = _name; // set new name
198                 nameToIdx[_name] = idx; // set new idx
199             }
200             if (getIdxByAddr(_ioOperatorAddr) == 0) { // not taken
201                 setAddrIdx(candidates[idx].ioOperatorAddr, 0);
202                 candidates[idx].ioOperatorAddr = _ioOperatorAddr;
203                 setAddrIdx(_ioOperatorAddr, idx);
204             }
205             if (getIdxByAddr(_ioRewardAddr) == 0) { // not taken
206                 setAddrIdx(candidates[idx].ioRewardAddr, 0);
207                 candidates[idx].ioRewardAddr = _ioRewardAddr;
208                 setAddrIdx(_ioRewardAddr, idx);
209             }
210         } else { // no entry, creating a new one.
211             require(nameToIdx[_name] == 0, "name taken");
212             require(getIdxByAddr(_ioOperatorAddr) == 0 && getIdxByAddr(_ioRewardAddr)
                    ==0, "ioAddr taken");
213             require(token.transferFrom(msg.sender, feeCollector, nameRegistrationFee),
                    "Fee required"); // transfer token to contract
214             // TODO: check if two addr are the same
215             idx = candidateCount;
216             candidateCount++; // prepare for the next one
217             candidates[idx].name = _name;
218             candidates[idx].addr = msg.sender;
219             candidates[idx].ioOperatorAddr = _ioOperatorAddr;
220             candidates[idx].ioRewardAddr = _ioRewardAddr;
221             candidates[idx].weight = 1;
222             nameToIdx[_name] = idx; // set reserve mapping
223             addrToIdx[msg.sender] = idx;
224             setAddrIdx(_ioOperatorAddr, idx);
225             setAddrIdx(_ioRewardAddr, idx);
226         }
227         emit Registered(idx, _name, msg.sender, _ioOperatorAddr, _ioRewardAddr, _data);
228     }
```

```
}
```

File Staking.sol

```
pragma solidity ^0.4.24;

import "./library/ERC20.sol";
import "./library/Ownable.sol";
import "./library/SafeMath.sol";
import "./library/Pausable.sol";
import "./library/Whitelist.sol";

/**
 * @title Staking and voting contract.
 * @author IoTeX Team
 *
 */
contract Staking is Pausable, Whitelist {
    using SafeMath for uint256;

    // Events to be emitted
    event BucketCreated(uint256 bucketIndex, bytes12 canName, uint256 amount, uint256 stakeDuration, bool nonDecay, bytes data);
    event BucketUpdated(uint256 bucketIndex, bytes12 canName, uint256 stakeDuration, uint256 stakeStartTime, bool nonDecay, address bucketOwner, bytes data);
    event BucketUnstake(uint256 bucketIndex, bytes12 canName, uint256 amount, bytes data);
    event BucketWithdraw(uint256 bucketIndex, bytes12 canName, uint256 amount, bytes data);
    // TODO add change owner event which is not covered by BucketUpdated event

    // IOTX used for staking
    ERC20 stakingToken;

    // Unit is epoch
    uint256 constant minStakeDuration = 0;
    uint256 constant maxStakeDuration = 350;
    uint256 constant minStakeAmount = 100 * 10 ** 18;
    uint256 constant unStakeDuration = 3;

    uint256 constant maxBucketsPerAddr = 500;
    uint256 constant secondsPerEpoch = 86400;

    // Core data structure to track staking/voting status
    struct Bucket {
        bytes12 canName;          // Candidate name, which maps to public keys by NameRegistration.sol
        uint256 stakedAmount;     // Number of tokens
        uint256 stakeDuration;    // Stake duration, unit: second since epoch
        uint256 stakeStartTime;   // Staking start time, unit: second since epoch
        bool nonDecay;            // Nondecay staking -- staking for N epochs consistently without decaying
        uint256 unstakeStartTime; // unstake timestamp, unit: second since epoch
        address bucketOwner;      // Owner of this bucket, usually the one who created it but can be someone else
        uint256 createTime;       // bucket firstly create time
        uint256 prev;             // Prev non-zero bucket index
        uint256 next;             // Next non-zero bucket index
    }
```

```
49      mapping(uint256 => Bucket) public buckets;
50      uint256 bucketCount; // number of total buckets. used to track the last used index
            for the bucket
51
52      // Map from owner address to array of bucket indexes.
53      mapping(address => uint256[]) public stakeholders;
54
55      /**
56       * @dev Modifier that checks that this given bucket can be updated/deleted by msg.
             sender
57       * @param _address address to transfer tokens from
58       * @param _bucketIndex uint256 the index of the bucket
59       */
60      modifier canTouchBucket(address _address, uint256 _bucketIndex) {
61          require(_address != address(0));
62          require(buckets[_bucketIndex].bucketOwner == msg.sender, "sender is not the
                owner.");
63          _;
64      }
65
66      /**
67       * @dev Modifier that check if a duration meets requirement
68       * @param _duration uint256 duration to check
69       */
70      modifier checkStakeDuration(uint256 _duration) {
71          require(_duration >= minStakeDuration && _duration <= maxStakeDuration, "The
                stake duration is too small or large");
72          require(_duration % 7 == 0, "The stake duration should be multiple of 7");
73          _;
74      }
75
76      /**
77       * @dev Constructor function
78       * @param _stakingTokenAddr address The address of the token contract used for
             staking
79       */
80      constructor(address _stakingTokenAddr) public {
81          stakingToken = ERC20(_stakingTokenAddr);
82          // create one bucket to initialize the double linked list
83          buckets[0] = Bucket("", 1, 0, block.timestamp, true, 0, msg.sender, block.
                timestamp, 0, 0);
84          stakeholders[msg.sender].push(0);
85          bucketCount = 1;
86      }
87
88      function getActiveBucketIdxImpl(uint256 _prevIndex, uint256 _limit) internal
            returns(uint256 count, uint256[] indexes) {
89          require (_limit > 0 && _limit < 5000);
90          Bucket memory bucket = buckets[_prevIndex];
91          //CTK: _prevIndex cannot be the last index/element in the buckets.
92          require(bucket.next > 0, "cannot find bucket based on input index.");
93
94          indexes = new uint256[](_limit);
95          uint256 i = 0;
96          for (i = 0; i < _limit; i++) {
97              while (bucket.next > 0 && buckets[bucket.next].unstakeStartTime > 0) { //
                    unstaked.
98                  bucket = buckets[bucket.next]; // skip
```

```
99              }
100             if (bucket.next == 0) { // no new bucket
101                 break;
102             }
103             indexes[i] = bucket.next;
104             bucket = buckets[bucket.next];
105         }
106         return (i, indexes);
107     }
108
109     function getActiveBucketIdx(uint256 _prevIndex, uint256 _limit) external view
            returns(uint256 count, uint256[] indexes) {
110         return getActiveBucketIdxImpl(_prevIndex, _limit);
111     }
112
113     /**
114      * @dev Get active buckets for a range of indexes
115      * @param _prevIndex uint256 the starting index. starting from 0, ending at the
             last. (putting 0,2 will return 1,2.)
116      * @param _limit uint256 the number of non zero buckets to fetch after the start
             index
117      * @return (uint256, uint256[], uint256[], uint256[], uint256[], bytes, address[])
118      *  count, index array, stakeStartTime array, duration array, decay array,
             stakedAmount array, concat stakedFor, ownerAddress array
119      */
120     function getActiveBuckets(uint256 _prevIndex, uint256 _limit) external view
            returns(uint256 count,
121             uint256[] indexes, uint256[] stakeStartTimes, uint256[] stakeDurations,
                bool[] decays, uint256[] stakedAmounts, bytes12[] canNames, address[]
                owners) {
122
123         (count, indexes) = getActiveBucketIdxImpl(_prevIndex, _limit);
124         //CTK: didn't check if count is 0.
125         stakeStartTimes = new uint256[](count);
126         stakeDurations = new uint256[](count);
127         decays = new bool[](count);
128         stakedAmounts = new uint256[](count);
129         canNames = new bytes12[](count);
130         owners = new address[](count);
131
132         for (uint256 i = 0; i < count; i++) {
133             Bucket memory bucket = buckets[indexes[i]];
134             stakeStartTimes[i] = bucket.stakeStartTime;
135             stakeDurations[i] = bucket.stakeDuration;
136             decays[i] = !bucket.nonDecay;
137             stakedAmounts[i] = bucket.stakedAmount;
138             canNames[i] = bucket.canName;
139             owners[i] = bucket.bucketOwner;
140
141         }
142
143         return (count, indexes, stakeStartTimes, stakeDurations, decays, stakedAmounts,
                canNames, owners);
144     }
145
146
147     function getActiveBucketCreateTimes(uint256 _prevIndex, uint256 _limit) external
            view returns(uint256 count,
```

```
148             uint256[] indexes, uint256[] createTimes) {
149         (count, indexes) = getActiveBucketIdxImpl(_prevIndex, _limit);
150         createTimes = new uint256[](count);
151         for (uint256 i = 0; i < count; i++) {
152             createTimes[i] = buckets[indexes[i]].createTime;
153         }
154         return (count, indexes, createTimes);
155     }
156 
157     /**
158      * @dev Get bucket indexes from a given address
159      * @param _owner address onwer of the buckets
160      * @return (uint256[])
161      */
162     /*@CTK getBucketIndexesByAddress
163       @post __reverted == false
164       @post __return == stakeholders[_owner]
165      */
166     function getBucketIndexesByAddress(address _owner) external view returns(uint256
            []) {
167         return stakeholders[_owner];
168     }
169 
170     /**
171      * @notice Extend the stake to stakeDuration from current time and/or set nonDecay
                .
172      * @notice MUST trigger BucketUpdated event
173      * @param _bucketIndex uint256 the index of the bucket
174      * @param _stakeDuration uint256 the desired duration of staking.
175      * @param _nonDecay bool if auto restake
176      * @param _data bytes optional data to include in the emitted event
177      */
178     /*@CTK restake
179       @tag assume_completion
180       @post __post.buckets[_bucketIndex].stakeDuration == _stakeDuration
181       @post __post.buckets[_bucketIndex].stakeStartTime == block.timestamp
182       @post __post.buckets[_bucketIndex].nonDecay == _nonDecay
183       @post __post.buckets[_bucketIndex].unstakeStartTime == 0
184      */
185     function restake(uint256 _bucketIndex, uint256 _stakeDuration, bool _nonDecay,
            bytes _data)
186             external whenNotPaused canTouchBucket(msg.sender, _bucketIndex)
                    checkStakeDuration(_stakeDuration) {
187         require(block.timestamp.add(_stakeDuration * secondsPerEpoch) >=
188                 buckets[_bucketIndex].stakeStartTime.add(buckets[_bucketIndex].
                        stakeDuration * secondsPerEpoch),
189                 "cannot reduce the stake duration.");
190         buckets[_bucketIndex].stakeDuration = _stakeDuration;
191         buckets[_bucketIndex].stakeStartTime = block.timestamp;
192         buckets[_bucketIndex].nonDecay = _nonDecay;
193         buckets[_bucketIndex].unstakeStartTime = 0;
194         emitBucketUpdated(_bucketIndex, _data);
195     }
196 
197     /*
198      * @notice Vote for another candidate with the tokens that are already staked in
             the given bucket
199      * @notice MUST trigger BucketUpdated event
```

```
200      * @param _bucketIndex uint256 the index of the bucket
201      * @param canName bytes the IoTeX address of the candidate the tokens are staked
             for
202      * @param _data bytes optional data to include in the emitted event
203      */
204     /*@CTK restake
205       @tag assume_completion
206       @post __post.buckets[_bucketIndex].canName == _canName
207       @post __post.buckets[_bucketIndex].unstakeStartTime == 0
208      */
209     function revote(uint256 _bucketIndex, bytes12 _canName, bytes _data)
210             external whenNotPaused canTouchBucket(msg.sender, _bucketIndex) {
211         require(buckets[_bucketIndex].unstakeStartTime == 0, "cannot revote during
                unstaking.");
212         buckets[_bucketIndex].canName = _canName;
213         emitBucketUpdated(_bucketIndex, _data);
214     }
215
216     /*
217      * @notice Set the new owner of a given bucket, the sender must be whitelisted to
             do so to avoid spam
218      * @notice MUST trigger BucketUpdated event
219      * @param _name bytes12 the name of the candidate the tokens are staked for
220      * @param _bucketIndex uint256 optional data to include in the Stake event
221      * @param _data bytes optional data to include in the emitted event
222      */
223     function setBucketOwner(uint256 _bucketIndex, address _newOwner, bytes _data)
224             external whenNotPaused onlyWhitelisted canTouchBucket(msg.sender,
                    _bucketIndex) {
225         removeBucketIndex(_bucketIndex);
226         buckets[_bucketIndex].bucketOwner = _newOwner;
227         stakeholders[_newOwner].push(_bucketIndex);
228         // TODO split event.
229         emitBucketUpdated(_bucketIndex, _data);
230     }
231
232     /**
233      * @notice Unstake a certain amount of tokens from a given bucket.
234      * @notice MUST trigger BucketUnstake event
235      * @param _bucketIndex uint256 the index of the bucket
236      * @param _data bytes optional data to include in the emitted event
237      */
238     /*@CTK unstake
239       @tag assume_completion
240       @pre secondsPerEpoch == 1
241       @post _bucketIndex > 0
242       @post !buckets[_bucketIndex].nonDecay
243       @post buckets[_bucketIndex].stakeStartTime + buckets[_bucketIndex].stakeDuration
              * secondsPerEpoch <= block.timestamp
244       @post __post.buckets[_bucketIndex].unstakeStartTime == block.timestamp
245      */
246     function unstake(uint256 _bucketIndex, bytes _data)
247             external whenNotPaused canTouchBucket(msg.sender, _bucketIndex) {
248         require(_bucketIndex > 0, "bucket 0 cannot be unstaked and withdrawn.");
249         require(!buckets[_bucketIndex].nonDecay, "Cannot unstake with nonDecay flag.
                Need to disable non-decay mode first.");
250         require(buckets[_bucketIndex].stakeStartTime.add(buckets[_bucketIndex].
                stakeDuration * secondsPerEpoch) <= block.timestamp,
```

```
251             "Staking time does not expire yet. Please wait until staking expires.");
252         require(buckets[_bucketIndex].unstakeStartTime == 0, "Unstaked already. No need
                to unstake again.");
253         buckets[_bucketIndex].unstakeStartTime = block.timestamp;
254         emit BucketUnstake(_bucketIndex, buckets[_bucketIndex].canName, buckets[
                _bucketIndex].stakedAmount, _data);
255     }
256
257     /**
258      * @notice this SHOULD return the given amount of tokens to the user, if unstaking
                is currently not possible the function MUST revert
259      * @notice MUST trigger BucketWithdraw event
260      * @param _bucketIndex uint256 the index of the bucket
261      * @param _data bytes optional data to include in the emitted event
262      */
263     function withdraw(uint256 _bucketIndex, bytes _data)
264             external whenNotPaused canTouchBucket(msg.sender, _bucketIndex) {
265         require(buckets[_bucketIndex].unstakeStartTime > 0, "Please unstake first
                before withdraw.");
266         require(
267             buckets[_bucketIndex].unstakeStartTime.add(unStakeDuration *
                    secondsPerEpoch) <= block.timestamp,
268             "Stakeholder needs to wait for 3 days before withdrawing tokens.");
269
270         // fix double linked list
271         uint256 prev = buckets[_bucketIndex].prev;
272         uint256 next = buckets[_bucketIndex].next;
273         buckets[prev].next = next;
274         buckets[next].prev = prev;
275
276         uint256 amount = buckets[_bucketIndex].stakedAmount;
277         bytes12 canName = buckets[_bucketIndex].canName;
278         address bucketowner = buckets[_bucketIndex].bucketOwner;
279         buckets[_bucketIndex].stakedAmount = 0;
280         removeBucketIndex(_bucketIndex);
281         delete buckets[_bucketIndex];
282
283         require(stakingToken.transfer(bucketowner, amount), "Unable to withdraw stake")
                ;
284         emit BucketWithdraw(_bucketIndex, canName, amount, _data);
285     }
286
287     /**
288      * @notice Returns the total of tokens staked from all addresses
289      * @return uint256 The number of tokens staked from all addresses
290      */
291     function totalStaked() public view returns (uint256) {
292         return stakingToken.balanceOf(this);
293     }
294
295     /**
296      * @notice Address of the token being used by the staking interface
297      * @return address The address of the ERC20 token used for staking
298      */
299     function token() public view returns(address) {
300         return stakingToken;
301     }
302
```

```
303     /**
304      * @notice Emit BucketUpdated event
305      */
306     function emitBucketUpdated(uint256 _bucketIndex, bytes _data) internal {
307         Bucket memory b = buckets[_bucketIndex];
308         emit BucketUpdated(_bucketIndex, b.canName, b.stakeDuration, b.stakeStartTime,
                b.nonDecay, b.bucketOwner, _data);
309     }
310
311     /**
312      * @dev Create a bucket and vote for a given canName.
313      * @param _canName bytes The IoTeX address of the candidate the stake is being
             created for
314      * @param _amount uint256 The duration to lock the tokens for
315      * @param _stakeDuration bytes the desired duration of the staking
316      * @param _nonDecay bool if auto restake
317      * @param _data bytes optional data to include in the emitted event
318      * @return uint236 the index of new bucket
319      */
320     function createBucket(bytes12 _canName, uint256 _amount, uint256 _stakeDuration,
            bool _nonDecay, bytes _data)
321             external whenNotPaused checkStakeDuration(_stakeDuration) returns (uint256)
                    {
322         require(_amount >= minStakeAmount, "amount should >= 100.");
323         require(stakeholders[msg.sender].length <= maxBucketsPerAddr, "One address can
                have up limited buckets");
324         require(stakingToken.transferFrom(msg.sender, this, _amount), "Stake required")
                ; // transfer token to contract
325         // add a new bucket to the end of buckets array and fix the double linked list.
326         buckets[bucketCount] = Bucket(_canName, _amount, _stakeDuration, block.
                timestamp, _nonDecay, 0, msg.sender, block.timestamp, buckets[0].prev, 0);
327         buckets[buckets[0].prev].next = bucketCount;
328         buckets[0].prev = bucketCount;
329         stakeholders[msg.sender].push(bucketCount);
330         bucketCount++;
331         emit BucketCreated(bucketCount-1, _canName, _amount, _stakeDuration, _nonDecay,
                _data);
332         return bucketCount-1;
333         /*
334             emit BucketCreated(bucketCount, _canName, _amount, _stakeDuration,
                    _nonDecay, _data);
335             return bucketCount++;
336          */
337     }
338
339     /**
340      * @dev Remove the bucket index from stakeholders map
341      * @param _bucketidx uint256 the bucket index
342      */
343     function removeBucketIndex(uint256 _bucketidx) internal {
344         address owner = buckets[_bucketidx].bucketOwner;
345         require(stakeholders[owner].length > 0, "Expect the owner has at least one
                bucket index");
346
347         uint256 i = 0;
348         for (i = i; i < stakeholders[owner].length; i++) {
349           if(stakeholders[owner][i] == _bucketidx) {
350                 break;
```

```
          }
        }
        for (i = i; i < stakeholders[owner].length - 1; i++) {
          stakeholders[owner][i] = stakeholders[owner][i + 1];
        }
        stakeholders[owner].length--;
    }
}
```

File library/Ownable.sol

```
pragma solidity ^0.4.24;

contract Ownable {

    address public owner;

    modifier onlyOwner {
        require(isOwner(msg.sender));
        _;
    }
    /*@CTK Ownable
      @post __post.owner == msg.sender
     */
    function Ownable() public {
        owner = msg.sender;
    }

    /*@CTK transferOwnership
      @tag assume_completion
      @post owner == msg.sender
      @post __post.owner == _newOwner
     */
    function transferOwnership(address _newOwner) public onlyOwner {
        owner = _newOwner;
    }

    /*@CTK isOwner
      @post (__post.owner == _address) -> __return
      @post (__post.owner != _address) -> !__return
     */
    function isOwner(address _address) public view returns (bool) {
        return owner == _address;
    }
}
```

File library/SafeMath.sol

```
pragma solidity ^0.4.24;

library SafeMath {

    /*@CTK SafeMath_mul
      @tag spec
      @post __reverted == __has_assertion_failure
      @post __has_assertion_failure == __has_overflow
      @post __reverted == false -> __return == a * b
      @post msg == msg__post
      @post (a > 0 && (a * b / a != b)) == __has_assertion_failure
      @post __addr_map == __addr_map__post
```

```
13      */
14      function mul(uint a, uint b) internal pure returns (uint) {
15          uint c = a * b;
16          assert(a == 0 || c / a == b);
17          return c;
18      }
19
20      /*@CTK SafeMath_div
21        @tag spec
22        @pre b != 0
23        @post __reverted == __has_assertion_failure
24        @post __has_overflow == true -> __has_assertion_failure == true
25        @post __reverted == false -> __return == a / b
26        @post msg == msg__post
27        @post (b == 0) == __has_assertion_failure
28        @post __addr_map == __addr_map__post
29       */
30      function div(uint a, uint b) internal pure returns (uint) {
31          assert(b > 0);
32          uint c = a / b;
33          assert(a == b * c + a % b);
34          return c;
35      }
36
37      /*@CTK SafeMath_sub
38        @tag spec
39        @post __reverted == __has_assertion_failure
40        @post __has_overflow == true -> __has_assertion_failure == true
41        @post __reverted == false -> __return == a - b
42        @post msg == msg__post
43        @post (b > a) == __has_assertion_failure
44        @post __addr_map == __addr_map__post
45       */
46      function sub(uint a, uint b) internal pure returns (uint) {
47          assert(b <= a);
48          return a - b;
49      }
50
51      /*@CTK SafeMath_add
52        @tag spec
53        @post __reverted == __has_assertion_failure
54        @post __has_assertion_failure == __has_overflow
55        @post __reverted == false -> __return == a + b
56        @post msg == msg__post
57        @post (a + b < a) == __has_assertion_failure
58        @post __addr_map == __addr_map__post
59       */
60      function add(uint a, uint b) internal pure returns (uint) {
61          uint c = a + b;
62          assert(c >= a);
63          return c;
64      }
65
66      /*@CTK "max64 case 1"
67        @tag spec
68        @pre a >= b
69        @post __return == a
70       */
```

```
71      /*@CTK "max64 case 2"
72        @tag spec
73        @pre a < b
74        @post __return == b
75       */
76      function max64(uint64 a, uint64 b) internal pure returns (uint64) {
77          return a >= b ? a : b;
78      }
79
80      /*@CTK "min64 case 1"
81        @tag spec
82        @pre a < b
83        @post __return == a
84       */
85      /*@CTK "min64 case 2"
86        @tag spec
87        @pre a >= b
88        @post __return == b
89       */
90      function min64(uint64 a, uint64 b) internal pure returns (uint64) {
91          return a < b ? a : b;
92      }
93
94      /*@CTK "max256 case 1"
95        @tag spec
96        @pre a >= b
97        @post __return == a
98       */
99      /*@CTK "max256 case 2"
100       @tag spec
101       @pre a < b
102       @post __return == b
103      */
104     function max256(uint a, uint b) internal pure returns (uint) {
105         return a >= b ? a : b;
106     }
107
108     /*@CTK "min256 case 1"
109       @tag spec
110       @pre a < b
111       @post __return == a
112      */
113     /*@CTK "min256 case 2"
114       @tag spec
115       @pre a >= b
116       @post __return == b
117      */
118     function min256(uint a, uint b) internal pure returns (uint) {
119         return a < b ? a : b;
120     }
121 }
```

File library/Whitelist.sol

```
1  pragma solidity ^0.4.24;
2
3
4  import "./Ownable.sol";
5
```

```
6
7  /**
8   * @title Whitelist
9   * @dev The Whitelist contract has a whitelist of addresses, and provides basic
        authorization control functions.
10  * @dev This simplifies the implementation of "user permissions".
11  */
12 contract Whitelist is Ownable {
13   mapping(address => bool) public whitelist;
14
15   event WhitelistedAddressAdded(address addr);
16   event WhitelistedAddressRemoved(address addr);
17
18   /**
19    * @dev Throws if called by any account that's not whitelisted.
20    */
21   modifier onlyWhitelisted() {
22     require(whitelist[msg.sender]);
23     _;
24   }
25
26   /**
27    * @dev add an address to the whitelist
28    * @param addr address
29    * @return true if the address was added to the whitelist, false if the address was
        already in the whitelist
30    */
31   /*@CTK "add address to whitelist"
32     @tag assume_completion
33     @post owner == msg.sender
34     @post __post.whitelist[addr] == true
35    */
36   function addAddressToWhitelist(address addr) onlyOwner public returns(bool success)
        {
37     if (!whitelist[addr]) {
38       whitelist[addr] = true;
39       emit WhitelistedAddressAdded(addr);
40       success = true;
41     }
42   }
43
44   /**
45    * @dev add addresses to the whitelist
46    * @param addrs addresses
47    * @return true if at least one address was added to the whitelist,
48    * false if all addresses were already in the whitelist
49    */
50   function addAddressesToWhitelist(address[] addrs) onlyOwner public returns(bool
        success) {
51     for (uint256 i = 0; i < addrs.length; i++) {
52       if (addAddressToWhitelist(addrs[i])) {
53         success = true;
54       }
55     }
56   }
57
58   /**
59    * @dev remove an address from the whitelist
```

```
60     * @param addr address
61     * @return true if the address was removed from the whitelist,
62     * false if the address wasn't in the whitelist in the first place
63     */
64    function removeAddressFromWhitelist(address addr) onlyOwner public returns(bool
          success) {
65      if (whitelist[addr]) {
66        whitelist[addr] = false;
67        emit WhitelistedAddressRemoved(addr);
68        success = true;
69      }
70    }
71
72    /**
73     * @dev remove addresses from the whitelist
74     * @param addrs addresses
75     * @return true if at least one address was removed from the whitelist,
76     * false if all addresses weren't in the whitelist in the first place
77     */
78    function removeAddressesFromWhitelist(address[] addrs) onlyOwner public returns(bool
           success) {
79      for (uint256 i = 0; i < addrs.length; i++) {
80        if (removeAddressFromWhitelist(addrs[i])) {
81          success = true;
82        }
83      }
84    }
85
86  }
```

File library/Pausable.sol

```
1  pragma solidity ^0.4.24;
2
3  import "./Ownable.sol";
4
5  /**
6   * @title Pausable
7   * @dev Base contract which allows children to implement an emergency stop mechanism.
8   */
9  contract Pausable is Ownable {
10   event Pause();
11   event Unpause();
12
13   bool public paused = false;
14
15
16   /**
17    * @dev Modifier to make a function callable only when the contract is not paused.
18    */
19   modifier whenNotPaused() {
20     require(!paused);
21     _;
22   }
23
24   /**
25    * @dev Modifier to make a function callable only when the contract is paused.
26    */
27   modifier whenPaused() {
```

```
28      require(paused);
29      _;
30    }
31
32    /**
33     * @dev called by the owner to pause, triggers stopped state
34     */
35    /*@CTK pause
36      @tag assume_completion
37      @post owner == msg.sender
38      @post !paused
39      @post __post.paused
40     */
41    function pause() onlyOwner whenNotPaused public {
42      paused = true;
43      emit Pause();
44    }
45
46    /**
47     * @dev called by the owner to unpause, returns to normal state
48     */
49    /*@CTK unpause
50      @tag assume_completion
51      @post owner == msg.sender
52      @post paused
53      @post !__post.paused
54     */
55    function unpause() onlyOwner whenPaused public {
56      paused = false;
57      emit Unpause();
58    }
59  }
```

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 IOTX.sol

```
1 pragma solidity ^0.4.23;
```

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

INSECURE_COMPILER_VERSION

Line 1 in File NameRegistration.sol

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

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

INSECURE_COMPILER_VERSION

Line 1 in File Staking.sol

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

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

TIMESTAMP_DEPENDENCY

Line 191 in File Staking.sol

```
191         buckets[_bucketIndex].stakeStartTime = block.timestamp;
```

"block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 250 in File Staking.sol

```
250         require(buckets[_bucketIndex].stakeStartTime.add(buckets[_bucketIndex].
                stakeDuration * secondsPerEpoch) <= block.timestamp,
```

"block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 253 in File Staking.sol

```
253         buckets[_bucketIndex].unstakeStartTime = block.timestamp;
```

"block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 267 in File Staking.sol

```
267             buckets[_bucketIndex].unstakeStartTime.add(unStakeDuration *
                    secondsPerEpoch) <= block.timestamp,
```

"block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 326 in File Staking.sol

```
326         buckets[bucketCount] = Bucket(_canName, _amount, _stakeDuration, block.
                timestamp, _nonDecay, 0, msg.sender, block.timestamp, buckets[0].prev, 0);
```

"block.timestamp" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 326 in File Staking.sol

```
326         buckets[bucketCount] = Bucket(_canName, _amount, _stakeDuration, block.
                timestamp, _nonDecay, 0, msg.sender, block.timestamp, buckets[0].prev, 0);
```

"block.timestamp" can be influenced by minors to some degree

INSECURE_COMPILER_VERSION

Line 1 in File Ownable.sol

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

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

INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

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

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

INSECURE_COMPILER_VERSION

Line 1 in File Whitelist.sol

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

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

INSECURE_COMPILER_VERSION

Line 1 in File Pausable.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_mul

04, Mar 2019
229.31 ms

Line 13-19 in File IOTX.sol

```
13  /*@CTK SafeMath_mul
14    @tag spec
15    @post __reverted == __has_assertion_failure
16    @post __has_assertion_failure == __has_overflow
17    @post __reverted == false -> c == a * b
18    @post msg == msg__post
19   */
```

Line 21-28 in File IOTX.sol

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

The code meets the specification

Formal Verification Request 2

SafeMath_div

04, Mar 2019
7.97 ms

Line 33-40 in File IOTX.sol

```
33  /*@CTK SafeMath_div
34    @tag spec
35    @pre b != 0
36    @post __reverted == __has_assertion_failure
37    @post __has_overflow == true -> __has_assertion_failure == true
38    @post __reverted == false -> __return == a / b
39    @post msg == msg__post
40   */
```

Line 42-47 in File IOTX.sol

```
42  function div(uint256 a, uint256 b) internal pure returns (uint256) {
43    // assert(b > 0); // Solidity automatically throws when dividing by 0
44    // uint256 c = a / b;
45    // assert(a == b * c + a % b); // There is no case in which this doesn't hold
46    return a / b;
47  }
```

The code meets the specification

Formal Verification Request 3

SafeMath_sub

04, Mar 2019
15.04 ms

Line 52-58 in File IOTX.sol

```
52  /*@CTK SafeMath_sub
53    @tag spec
54    @post __reverted == __has_assertion_failure
55    @post __has_overflow == true -> __has_assertion_failure == true
56    @post __reverted == false -> __return == a - b
57    @post msg == msg__post
58   */
```

Line 60-63 in File IOTX.sol

```
60  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
61    assert(b <= a);
62    return a - b;
63  }
```

The code meets the specification

Formal Verification Request 4

SafeMath_add

04, Mar 2019
17.82 ms

Line 68-74 in File IOTX.sol

```
68  /*@CTK SafeMath_add
69    @tag spec
70    @post __reverted == __has_assertion_failure
71    @post __has_assertion_failure == __has_overflow
72    @post __reverted == false -> c == a + b
73    @post msg == msg__post
74   */
```

Line 76-80 in File IOTX.sol

```
76  function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
77    c = a + b;
78    assert(c >= a);
79    return c;
80  }
```

The code meets the specification

Formal Verification Request 5

owner_set_on_success

04, Mar 2019
5.41 ms

Line 99-101 in File IOTX.sol

```
99  /*@CTK owner_set_on_success
100   @pre __reverted == false -> __post.owner == owner
101  */
```

Line 103-105 in File IOTX.sol

```
103 function Ownable() public {
104   owner = msg.sender;
105 }
```

The code meets the specification

Formal Verification Request 6

transferOwnership

04, Mar 2019
25.33 ms

Line 119-123 in File IOTX.sol

```
119 /*@CTK transferOwnership
120   @post __reverted == false -> (msg.sender == owner -> __post.owner == newOwner)
121   @post (owner != msg.sender) -> (__reverted == true)
122   @post (newOwner == address(0)) -> (__reverted == true)
123  */
```

Line 125-129 in File IOTX.sol

```
125 function transferOwnership(address newOwner) public onlyOwner {
126   require(newOwner != address(0));
127   emit OwnershipTransferred(owner, newOwner);
128   owner = newOwner;
129 }
```

The code meets the specification

Formal Verification Request 7

transfer_success

04, Mar 2019
45.71 ms

Line 224-228 in File IOTX.sol

```
/*@CTK transfer_success
  @tag assume_completion
  @post _to != address(0)
  @post balances[msg.sender] >= _value
 */
```

Line 236-244 in File IOTX.sol

```
function transfer(address _to, uint256 _value) public returns (bool) {
  require(_to != address(0));
  require(_value <= balances[msg.sender]);

  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 8

balanceOf

04, Mar 2019
5.96 ms

Line 251-254 in File IOTX.sol

```
/*@CTK balanceOf
  @post __reverted == false
  @post __return == balances[_owner]
 */
```

Line 256-258 in File IOTX.sol

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

The code meets the specification

Formal Verification Request 9

approve_success

04, Mar 2019
9.7 ms

Line 309-312 in File IOTX.sol

```
/*@CTK approve_success
  @post _value == 0 -> __reverted == false
  @post allowed[msg.sender][_spender] == 0 -> __reverted == false
 */
```

Line 318-322 in File IOTX.sol

```
318  function approve(address _spender, uint256 _value) public returns (bool) {
319    allowed[msg.sender][_spender] = _value;
320    emit Approval(msg.sender, _spender, _value);
321    return true;
322  }
```

The code meets the specification

Formal Verification Request 10

approve

04, Mar 2019
1.31 ms

Line 313-316 in File IOTX.sol

```
313  /*@CTK approve
314    @tag assume_completion
315    @post __post.allowed[msg.sender][_spender] == _value
316   */
```

Line 318-322 in File IOTX.sol

```
318  function approve(address _spender, uint256 _value) public returns (bool) {
319    allowed[msg.sender][_spender] = _value;
320    emit Approval(msg.sender, _spender, _value);
321    return true;
322  }
```

The code meets the specification

Formal Verification Request 11

CtkIncreaseApprovalEffect

04, Mar 2019
20.06 ms

Line 344-348 in File IOTX.sol

```
344  /*@CTK CtkIncreaseApprovalEffect
345    @tag assume_completion
346    @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
         _addedValue
347    @post __has_overflow == false
348   */
```

Line 350-354 in File IOTX.sol

```
350  function increaseApproval(address _spender, uint _addedValue) public returns (bool)
       {
351    allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
352    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
```

```
353     return true;
354   }
```

The code meets the specification

Formal Verification Request 12

CtkDecreaseApprovalEffect_1

04, Mar 2019
30.63 ms

Line 366-371 in File IOTX.sol

```
366   /*@CTK CtkDecreaseApprovalEffect_1
367     @pre allowed[msg.sender][_spender] >= _subtractedValue
368     @tag assume_completion
369     @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
          _subtractedValue
370     @post __has_overflow == false
371    */
```

Line 379-388 in File IOTX.sol

```
379   function decreaseApproval(address _spender, uint _subtractedValue) public returns (
          bool) {
380     uint oldValue = allowed[msg.sender][_spender];
381     if (_subtractedValue > oldValue) {
382       allowed[msg.sender][_spender] = 0;
383     } else {
384       allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
385     }
386     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
387     return true;
388   }
```

The code meets the specification

Formal Verification Request 13

CtkDecreaseApprovalEffect_2

04, Mar 2019
1.8 ms

Line 372-377 in File IOTX.sol

```
372   /*@CTK CtkDecreaseApprovalEffect_2
373     @pre allowed[msg.sender][_spender] < _subtractedValue
374     @tag assume_completion
375     @post __post.allowed[msg.sender][_spender] == 0
376     @post __has_overflow == false
377    */
```

Line 379-388 in File IOTX.sol

```
379  function decreaseApproval(address _spender, uint _subtractedValue) public returns (
         bool) {
380    uint oldValue = allowed[msg.sender][_spender];
381    if (_subtractedValue > oldValue) {
382      allowed[msg.sender][_spender] = 0;
383    } else {
384      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
385    }
386    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
387    return true;
388  }
```

The code meets the specification

Formal Verification Request 14

CtkTransferFromNoEffect

04, Mar 2019
225.36 ms

Line 428-430 in File IOTX.sol

```
428    /*@CTK CtkTransferFromNoEffect
429      @post (_to == address(0)) \/ (paused == true) -> __reverted == true
430     */
```

Line 439-443 in File IOTX.sol

```
439    function transferFrom(address _from, address _to, uint _value) whenNotPaused
440        validDestination(_to)
441        returns (bool) {
442        return super.transferFrom(_from, _to, _value);
443    }
```

The code meets the specification

Formal Verification Request 15

CtkApproveNoEffect

04, Mar 2019
52.22 ms

Line 445-447 in File IOTX.sol

```
445    /*@CTK CtkApproveNoEffect
446      @post (paused == true) -> __post == this
447     */
```

Line 453-456 in File IOTX.sol

```
453    function approve(address _spender, uint256 _value) public whenNotPaused
454      returns (bool) {
455      return super.approve(_spender, _value);
456    }
```

The code meets the specification

Formal Verification Request 16

CtkApprove

04, Mar 2019
3.27 ms

Line 448-451 in File IOTX.sol

```
448     /*@CTK CtkApprove
449       @tag assume_completion
450       @post __post.allowed[msg.sender][_spender] == _value
451      */
```

Line 453-456 in File IOTX.sol

```
453     function approve(address _spender, uint256 _value) public whenNotPaused
454       returns (bool) {
455       return super.approve(_spender, _value);
456     }
```

The code meets the specification

Formal Verification Request 17

CtkIncreaseApprovalNoEffect

04, Mar 2019
69.15 ms

Line 458-460 in File IOTX.sol

```
458     /*@CTK CtkIncreaseApprovalNoEffect
459       @post (paused == true) -> __reverted == true
460      */
```

Line 468-471 in File IOTX.sol

```
468     function increaseApproval(address _spender, uint _addedValue) public whenNotPaused
469       returns (bool success) {
470       return super.increaseApproval(_spender, _addedValue);
471     }
```

The code meets the specification

Formal Verification Request 18

CtkIncreaseApprovalEffect

04, Mar 2019
3.94 ms

Line 461-466 in File IOTX.sol

```
461   /*@CTK CtkIncreaseApprovalEffect
462     @pre paused == false
463     @tag assume_completion
464     @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
          _addedValue
465     @post __has_overflow == false
466    */
```

Line 468-471 in File IOTX.sol

```
468   function increaseApproval(address _spender, uint _addedValue) public whenNotPaused
469     returns (bool success) {
470     return super.increaseApproval(_spender, _addedValue);
471   }
```

The code meets the specification

Formal Verification Request 19

CtkDecreaseApprovalNoEffect

04, Mar 2019
96.73 ms

Line 473-475 in File IOTX.sol

```
473   /*@CTK CtkDecreaseApprovalNoEffect
474     @post (paused == true) -> __reverted == true
475    */
```

Line 483-486 in File IOTX.sol

```
483   function decreaseApproval(address _spender, uint _subtractedValue) public
          whenNotPaused
484     returns (bool success) {
485     return super.decreaseApproval(_spender, _subtractedValue);
486   }
```

The code meets the specification

Formal Verification Request 20

CtkDecreaseApprovalEffect

04, Mar 2019
66.96 ms

Line 476-481 in File IOTX.sol

```
476   /*@CTK CtkDecreaseApprovalEffect
477     @pre allowed[msg.sender][_spender] >= _subtractedValue
478     @tag assume_completion
```

```
479     @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
            _subtractedValue
480     @post __has_overflow == false
481    */
```

Line 483-486 in File IOTX.sol

```
483   function decreaseApproval(address _spender, uint _subtractedValue) public
          whenNotPaused
484     returns (bool success) {
485     return super.decreaseApproval(_spender, _subtractedValue);
486   }
```

The code meets the specification

Formal Verification Request 21

NameRegistration

04, Mar 2019
12.18 ms

Line 38-40 in File NameRegistration.sol

```
38    /*@CTK NameRegistration
39      @post __post.candidateCount == 1
40     */
```

Line 41-44 in File NameRegistration.sol

```
41    constructor(address _tokenAddr) public {
42        token = ERC20(_tokenAddr);
43        candidateCount = 1; // 0 is null, reserved.
44    }
```

The code meets the specification

Formal Verification Request 22

setFeeCollector

04, Mar 2019
34.6 ms

Line 127-131 in File NameRegistration.sol

```
127   /*@CTK setFeeCollector
128     @tag assume_completion
129     @post owner == msg.sender
130     @post __post.feeCollector == _addr
131    */
```

Line 132-134 in File NameRegistration.sol

```
132   function setFeeCollector(address _addr) external onlyOwner {
133       feeCollector = _addr;
134   }
```

The code meets the specification

Formal Verification Request 23

setNameRegistrationFee

04, Mar 2019
29.07 ms

Line 136-140 in File NameRegistration.sol

```
136     /*@CTK setNameRegistrationFee
137       @tag assume_completion
138       @post owner == msg.sender
139       @post __post.nameRegistrationFee == _fee
140      */
```

Line 141-143 in File NameRegistration.sol

```
141     function setNameRegistrationFee(uint256 _fee) external onlyOwner {
142         nameRegistrationFee = _fee;
143     }
```

The code meets the specification

Formal Verification Request 24

setWeight

04, Mar 2019
48.72 ms

Line 145-150 in File NameRegistration.sol

```
145     /*@CTK setWeight
146       @tag assume_completion
147       @post owner == msg.sender
148       @post nameToIdx[_name] > 0
149       @post __post.candidates[nameToIdx[_name]].weight == _weight
150      */
```

Line 151-155 in File NameRegistration.sol

```
151     function setWeight(bytes12 _name, uint256 _weight) external onlyOwner {
152         uint256 idx = nameToIdx[_name];
153         require(idx > 0, "name not registered.");
154         candidates[idx].weight = _weight;
155     }
```

The code meets the specification

Formal Verification Request 25

setNameAddress

📅 04, Mar 2019
⏱ 81.6 ms

Line 157-164 in File NameRegistration.sol

```
157     /*@CTK setNameAddress
158       @tag assume_completion
159       @post owner == msg.sender
160       @post addrToIdx[_addr] == 0
161       @post candidates[nameToIdx[_name]].addr != _addr
162       @post __post.candidates[nameToIdx[_name]].addr == _addr
163       @post __post.addrToIdx[_addr] == nameToIdx[_name]
164      */
```

Line 166-174 in File NameRegistration.sol

```
166     function setNameAddress(bytes12 _name, address _addr) external onlyOwner {
167         require(addrToIdx[_addr] == 0, "new addr should not have name");
168         uint256 idx = nameToIdx[_name]; // find the candidate
169         address oldAddr = candidates[idx].addr;
170         require(oldAddr != _addr, "new address is expected");
171         candidates[idx].addr = _addr;
172         addrToIdx[_addr] = idx;
173         delete addrToIdx[oldAddr];
174     }
```

✅ The code meets the specification

Formal Verification Request 26

getBucketIndexesByAddress

📅 04, Mar 2019
⏱ 6.75 ms

Line 162-165 in File Staking.sol

```
162     /*@CTK getBucketIndexesByAddress
163       @post __reverted == false
164       @post __return == stakeholders[_owner]
165      */
```

Line 166-168 in File Staking.sol

```
166     function getBucketIndexesByAddress(address _owner) external view returns(uint256
            []) {
167         return stakeholders[_owner];
168     }
```

✅ The code meets the specification

Formal Verification Request 27

restake

04, Mar 2019
248.88 ms

Line 178-184 in File Staking.sol

```
178     /*@CTK restake
179       @tag assume_completion
180       @post __post.buckets[_bucketIndex].stakeDuration == _stakeDuration
181       @post __post.buckets[_bucketIndex].stakeStartTime == block.timestamp
182       @post __post.buckets[_bucketIndex].nonDecay == _nonDecay
183       @post __post.buckets[_bucketIndex].unstakeStartTime == 0
184      */
```

Line 185-195 in File Staking.sol

```
185     function restake(uint256 _bucketIndex, uint256 _stakeDuration, bool _nonDecay,
            bytes _data)
186             external whenNotPaused canTouchBucket(msg.sender, _bucketIndex)
                    checkStakeDuration(_stakeDuration) {
187         require(block.timestamp.add(_stakeDuration * secondsPerEpoch) >=
188                 buckets[_bucketIndex].stakeStartTime.add(buckets[_bucketIndex].
                        stakeDuration * secondsPerEpoch),
189                 "cannot reduce the stake duration.");
190         buckets[_bucketIndex].stakeDuration = _stakeDuration;
191         buckets[_bucketIndex].stakeStartTime = block.timestamp;
192         buckets[_bucketIndex].nonDecay = _nonDecay;
193         buckets[_bucketIndex].unstakeStartTime = 0;
194         emitBucketUpdated(_bucketIndex, _data);
195     }
```

The code meets the specification

Formal Verification Request 28

restake

04, Mar 2019
160.91 ms

Line 204-208 in File Staking.sol

```
204     /*@CTK restake
205       @tag assume_completion
206       @post __post.buckets[_bucketIndex].canName == _canName
207       @post __post.buckets[_bucketIndex].unstakeStartTime == 0
208      */
```

Line 209-214 in File Staking.sol

```
209     function revote(uint256 _bucketIndex, bytes12 _canName, bytes _data)
210             external whenNotPaused canTouchBucket(msg.sender, _bucketIndex) {
211         require(buckets[_bucketIndex].unstakeStartTime == 0, "cannot revote during
                unstaking.");
```

```
212         buckets[_bucketIndex].canName = _canName;
213         emitBucketUpdated(_bucketIndex, _data);
214     }
```

The code meets the specification

Formal Verification Request 29

unstake

04, Mar 2019
273.02 ms

Line 238-245 in File Staking.sol

```
238     /*@CTK unstake
239       @tag assume_completion
240       @pre secondsPerEpoch == 1
241       @post _bucketIndex > 0
242       @post !buckets[_bucketIndex].nonDecay
243       @post buckets[_bucketIndex].stakeStartTime + buckets[_bucketIndex].stakeDuration
              * secondsPerEpoch <= block.timestamp
244       @post __post.buckets[_bucketIndex].unstakeStartTime == block.timestamp
245      */
```

Line 246-255 in File Staking.sol

```
246     function unstake(uint256 _bucketIndex, bytes _data)
247             external whenNotPaused canTouchBucket(msg.sender, _bucketIndex) {
248         require(_bucketIndex > 0, "bucket 0 cannot be unstaked and withdrawn.");
249         require(!buckets[_bucketIndex].nonDecay, "Cannot unstake with nonDecay flag.
                Need to disable non-decay mode first.");
250         require(buckets[_bucketIndex].stakeStartTime.add(buckets[_bucketIndex].
                stakeDuration * secondsPerEpoch) <= block.timestamp,
251             "Staking time does not expire yet. Please wait until staking expires.");
252         require(buckets[_bucketIndex].unstakeStartTime == 0, "Unstaked already. No need
                 to unstake again.");
253         buckets[_bucketIndex].unstakeStartTime = block.timestamp;
254         emit BucketUnstake(_bucketIndex, buckets[_bucketIndex].canName, buckets[
                _bucketIndex].stakedAmount, _data);
255     }
```

The code meets the specification

Formal Verification Request 30

Ownable

04, Mar 2019
5.53 ms

Line 11-13 in File Ownable.sol

```
11    /*@CTK Ownable
12      @post __post.owner == msg.sender
13     */
```

Line 14-16 in File Ownable.sol

```
14    function Ownable() public {
15        owner = msg.sender;
16    }
```

✅ The code meets the specification

Formal Verification Request 31

transferOwnership

📅 04, Mar 2019
⏱ 32.41 ms

Line 18-22 in File Ownable.sol

```
18    /*@CTK transferOwnership
19      @tag assume_completion
20      @post owner == msg.sender
21      @post __post.owner == _newOwner
22     */
```

Line 23-25 in File Ownable.sol

```
23    function transferOwnership(address _newOwner) public onlyOwner {
24        owner = _newOwner;
25    }
```

✅ The code meets the specification

Formal Verification Request 32

isOwner

📅 04, Mar 2019
⏱ 0.41 ms

Line 27-30 in File Ownable.sol

```
27    /*@CTK isOwner
28      @post (__post.owner == _address) -> __return
29      @post (__post.owner != _address) -> !__return
30     */
```

Line 31-33 in File Ownable.sol

```
31    function isOwner(address _address) public view returns (bool) {
32        return owner == _address;
33    }
```

✅ The code meets the specification

Formal Verification Request 33

SafeMath_mul

04, Mar 2019
112.55 ms

Line 5-13 in File SafeMath.sol

```
5   /*@CTK SafeMath_mul
6     @tag spec
7     @post __reverted == __has_assertion_failure
8     @post __has_assertion_failure == __has_overflow
9     @post __reverted == false -> __return == a * b
10    @post msg == msg__post
11    @post (a > 0 && (a * b / a != b)) == __has_assertion_failure
12    @post __addr_map == __addr_map__post
13   */
```

Line 14-18 in File SafeMath.sol

```
14  function mul(uint a, uint b) internal pure returns (uint) {
15      uint c = a * b;
16      assert(a == 0 || c / a == b);
17      return c;
18  }
```

The code meets the specification

Formal Verification Request 34

SafeMath_div

04, Mar 2019
1141.48 ms

Line 20-29 in File SafeMath.sol

```
20  /*@CTK SafeMath_div
21    @tag spec
22    @pre b != 0
23    @post __reverted == __has_assertion_failure
24    @post __has_overflow == true -> __has_assertion_failure == true
25    @post __reverted == false -> __return == a / b
26    @post msg == msg__post
27    @post (b == 0) == __has_assertion_failure
28    @post __addr_map == __addr_map__post
29   */
```

Line 30-35 in File SafeMath.sol

```
30  function div(uint a, uint b) internal pure returns (uint) {
31      assert(b > 0);
32      uint c = a / b;
33      assert(a == b * c + a % b);
34      return c;
35  }
```

The code meets the specification

Formal Verification Request 35

SafeMath_sub

04, Mar 2019
14.63 ms

Line 37-45 in File SafeMath.sol

```
37    /*@CTK SafeMath_sub
38      @tag spec
39      @post __reverted == __has_assertion_failure
40      @post __has_overflow == true -> __has_assertion_failure == true
41      @post __reverted == false -> __return == a - b
42      @post msg == msg__post
43      @post (b > a) == __has_assertion_failure
44      @post __addr_map == __addr_map__post
45     */
```

Line 46-49 in File SafeMath.sol

```
46    function sub(uint a, uint b) internal pure returns (uint) {
47        assert(b <= a);
48        return a - b;
49    }
```

The code meets the specification

Formal Verification Request 36

SafeMath_add

04, Mar 2019
17.2 ms

Line 51-59 in File SafeMath.sol

```
51    /*@CTK SafeMath_add
52      @tag spec
53      @post __reverted == __has_assertion_failure
54      @post __has_assertion_failure == __has_overflow
55      @post __reverted == false -> __return == a + b
56      @post msg == msg__post
57      @post (a + b < a) == __has_assertion_failure
58      @post __addr_map == __addr_map__post
59     */
```

Line 60-64 in File SafeMath.sol

```
60    function add(uint a, uint b) internal pure returns (uint) {
61        uint c = a + b;
62        assert(c >= a);
63        return c;
64    }
```

The code meets the specification

Formal Verification Request 37

max64 case 1

04, Mar 2019
5.88 ms

Line 66-70 in File SafeMath.sol

```
66    /*@CTK "max64 case 1"
67      @tag spec
68      @pre a >= b
69      @post __return == a
70     */
```

Line 76-78 in File SafeMath.sol

```
76    function max64(uint64 a, uint64 b) internal pure returns (uint64) {
77        return a >= b ? a : b;
78    }
```

The code meets the specification

Formal Verification Request 38

max64 case 2

04, Mar 2019
0.65 ms

Line 71-75 in File SafeMath.sol

```
71    /*@CTK "max64 case 2"
72      @tag spec
73      @pre a < b
74      @post __return == b
75     */
```

Line 76-78 in File SafeMath.sol

```
76    function max64(uint64 a, uint64 b) internal pure returns (uint64) {
77        return a >= b ? a : b;
78    }
```

The code meets the specification

Formal Verification Request 39

min64 case 1

04, Mar 2019
5.88 ms

Line 80-84 in File SafeMath.sol

```
80    /*@CTK "min64 case 1"
81      @tag spec
82      @pre a < b
83      @post __return == a
84     */
```

Line 90-92 in File SafeMath.sol

```
90    function min64(uint64 a, uint64 b) internal pure returns (uint64) {
91        return a < b ? a : b;
92    }
```

The code meets the specification

Formal Verification Request 40

min64 case 2

04, Mar 2019
0.63 ms

Line 85-89 in File SafeMath.sol

```
85    /*@CTK "min64 case 2"
86      @tag spec
87      @pre a >= b
88      @post __return == b
89     */
```

Line 90-92 in File SafeMath.sol

```
90    function min64(uint64 a, uint64 b) internal pure returns (uint64) {
91        return a < b ? a : b;
92    }
```

The code meets the specification

Formal Verification Request 41

max256 case 1

04, Mar 2019
6.61 ms

Line 94-98 in File SafeMath.sol

```
94    /*@CTK "max256 case 1"
95      @tag spec
96      @pre a >= b
97      @post __return == a
98     */
```

Line 104-106 in File SafeMath.sol

```
104     function max256(uint a, uint b) internal pure returns (uint) {
105         return a >= b ? a : b;
106     }
```

The code meets the specification

Formal Verification Request 42

max256 case 2

04, Mar 2019
0.87 ms

Line 99-103 in File SafeMath.sol

```
99      /*@CTK "max256 case 2"
100       @tag spec
101       @pre a < b
102       @post __return == b
103     */
```

Line 104-106 in File SafeMath.sol

```
104     function max256(uint a, uint b) internal pure returns (uint) {
105         return a >= b ? a : b;
106     }
```

The code meets the specification

Formal Verification Request 43

min256 case 1

04, Mar 2019
6.2 ms

Line 108-112 in File SafeMath.sol

```
108     /*@CTK "min256 case 1"
109       @tag spec
110       @pre a < b
111       @post __return == a
112     */
```

Line 118-120 in File SafeMath.sol

```
118     function min256(uint a, uint b) internal pure returns (uint) {
119         return a < b ? a : b;
120     }
```

The code meets the specification

Formal Verification Request 44

min256 case 2

04, Mar 2019
1.31 ms

Line 113-117 in File SafeMath.sol

```
113   /*@CTK "min256 case 2"
114     @tag spec
115     @pre a >= b
116     @post __return == b
117    */
```

Line 118-120 in File SafeMath.sol

```
118   function min256(uint a, uint b) internal pure returns (uint) {
119       return a < b ? a : b;
120   }
```

The code meets the specification

Formal Verification Request 45

add address to whitelist

04, Mar 2019
40.81 ms

Line 31-35 in File Whitelist.sol

```
31  /*@CTK "add address to whitelist"
32    @tag assume_completion
33    @post owner == msg.sender
34    @post __post.whitelist[addr] == true
35   */
```

Line 36-42 in File Whitelist.sol

```
36  function addAddressToWhitelist(address addr) onlyOwner public returns(bool success)
        {
37    if (!whitelist[addr]) {
38      whitelist[addr] = true;
39      emit WhitelistedAddressAdded(addr);
40      success = true;
41    }
42  }
```

The code meets the specification

Formal Verification Request 46

pause

04, Mar 2019
48.92 ms

Line 35-40 in File Pausable.sol

```
35  /*@CTK pause
36    @tag assume_completion
37    @post owner == msg.sender
38    @post !paused
39    @post __post.paused
40   */
```

Line 41-44 in File Pausable.sol

```
41  function pause() onlyOwner whenNotPaused public {
42    paused = true;
43    emit Pause();
44  }
```

The code meets the specification

Formal Verification Request 47

unpause

04, Mar 2019
39.91 ms

Line 49-54 in File Pausable.sol

```
49  /*@CTK unpause
50    @tag assume_completion
51    @post owner == msg.sender
52    @post paused
53    @post !__post.paused
54   */
```

Line 55-58 in File Pausable.sol

```
55  function unpause() onlyOwner whenPaused public {
56    paused = false;
57    emit Unpause();
58  }
```

The code meets the specification