CERTIK

Audit Report

Produced by CertiK

for

20th Jan, 2020

CertiK Audit Report For Hiblocks

Request Date: 2019-01-15
Revision Date: 2020-01-20
Platform Name: Ethereum

CERTIK

Contents

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and Hiblocks(the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the "Agreement"). This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 6.2B in assets.

For more information: https://certik.org/

Executive Summary

This report has been prepared for Hiblocks to discover issues and vulnerabilities in the source code of their HiblocksToken.sol, ERC1132.sol, AdminRole.sol, HiblocksIERC20.sol, LockableToken.sol, PausableToken.sol and StakingToken.sol smart contracts. A comprehensive examination has been performed, utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Vulnerability Classification

CertiK categorizes issues into three buckets based on overall risk levels:

Code implementation does not match specification, which could result in the loss of funds for contract owner or users.

Code implementation does not match the specification under certain conditions, which could affect the security standard by loss of access control.

Code implementation does not follow best practices, or uses suboptimal design patterns, which could lead to security vulnerabilities further down the line.

Testing Summary

PASS

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

Jan 20, 2020

Type of Issues

CertiK's smart label engine applied 100% formal verification coverage on the source code. Our team of engineers has scanned the source code using proprietary static analysis tools and code-review methodologies. The following technical issues were found:

Title	Description	Issues	SWC ID
Integer Overflow/ Underflow	An overflow/underflow occurs when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function Incorrectness	Function implementation does not meet specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker can 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 miners to some degree.	1	SWC-116
Insecure Compiler Version	Using a 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	Using block attributes to generate random numbers is unreliable, as they can be influenced by miners 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

Title	Description	Issues	SWC ID
Delegatecall to Untrusted Callee	Calling untrusted contracts is very dangerous, so 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, meaning a malicious user can 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.	0	SWC-111
Unused Variables	Unused variables reduce code quality	0	SWC-131

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Source Code SHA-256 Checksum

- **HiblocksToken.sol**
  `7d7e96b7f1d92e42124d46d826e5e4041010ea3c9fb72f7a4a4ea01c0b6c9689`
- **RewardPool.sol**
  `bd4aede74b4641b0635bfdc2e8f603642b7722831c4e1bbd92a286ac28739690`
- **RewardSupplier.sol**
  `d85e6d1d33b1f7372bf7666f8add4398f881d72a3f23d0d835f7cefd37f9e759`
- **ERC1132.sol**
  `4ad6d50e9c529b639d347131a3938b7a4841dc7366958603311971b8f8b8de6f`
- **RewardSupplyForTest.sol**
  `0b53be40b5b356aafdb5e6026c3f687705c63d2079f07799fe8456bdceb9e5f3`
- **HiblocksIERC20.sol**
  `34a25ed4d653a5972a9d476f21d4ce9960698282edf4d0fc1dec4a2ffafc365e`
- **LockableToken.sol**
  `fd218a5f32006a3e4dbad7227b50085ff835a0ad4c99990e28d55990585da72b`
- **PausableToken.sol**
  `336bec2eba0d75f1e58137eb8a374b9fffb24ee98896beeef82a6a2c03df1030`
- **StakingToken.sol**
  `71c9bddc6fc6ce70158458c5e17f070930d27abe649e019d33315dc962dc94f9`
- **AdminRole.sol**
  `71562d45b379e0f57293d1f88d30ad5060a6f6434b6fa9d7384ddb0e448e565f`

Summary

CertiK worked closely with Hiblocks to audit the design and implementation of its soon-to-be released smart contract. To ensure comprehensive protection, the source code was analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with best practices.

Our client Hiblocks has demonstrated their professional and knowledgeable understanding of the project Hiblocks, by having 1) a production ready repository with high-quality source code; 2) unit tests covering the majority of its business scenarios; 3) accessible, clean, and accurate readme documents for intentions, functionalities, and responsibilities of the smart contracts.

Overall, we found Hiblocks's smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend

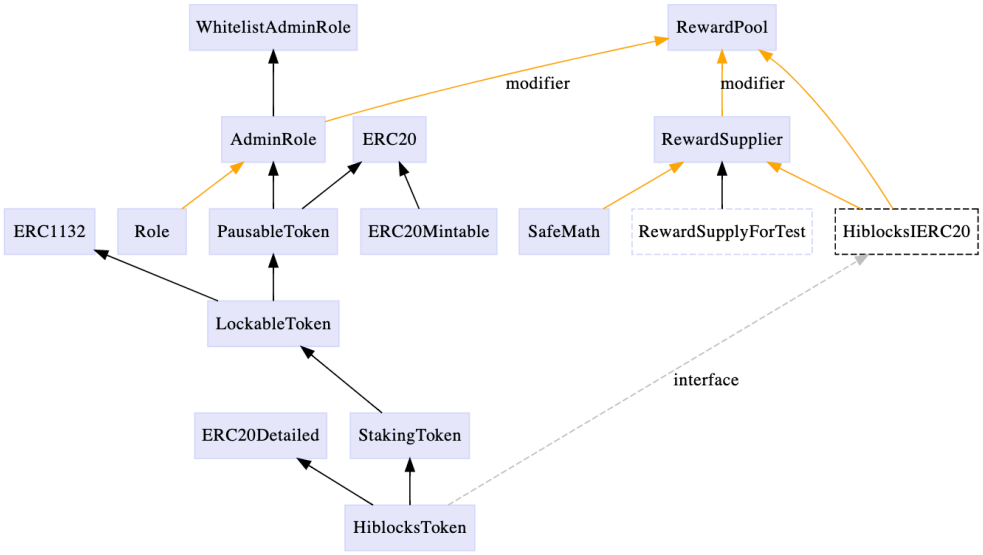
to seek multiple opinions, continually improve the codebase, and perform additional tests before the mainnet release.

Recommendations

Items in this section are not critical to the overall functionality of Hiblocks's smart contracts; however, we leave it to the client's discretion to decide whether to address them before the final deployment of source codes. Recommendations are labeled CRITICAL, MAJOR, MINOR, INFO, and DISCUSSION in decreasing significance level.

Design Architect & Analysis

Inheritance



Global State Variables

ERC1132.sol

Variable Name	Constant	Initialized By Func	Modified By Func
`lockReason`		default	
`locked`		default	

AdminRole.sol

Variable Name	Constant	Initialized By Func	Modified By Func
`_adminList`		default	

Paused.sol

Variable Name	Constant	Initialized By Func	Modified By Func
`_paused`		constructor	

StakingToken.sol

Variable Name	Constant	Initialized By Func	Modified By Func
`stakeholders`		constructor	
`secondsInterval`		constructor	
`validStakeDuration`		constructor	
`validStakeStartTime`		constructor	
`stakeFlag`		constructor	

HiblocksToken.sol

Variable Name	Constant	Initialized By Func	Modified By Func
`memos`			`transferWithMemo`

RewardPool.sol

Variable Name	Constant	Initialized By Func	Modified By Func
`rewardToken`			`payReward`

RewardSupplier.sol

Variable Name	Constant	Initialized By Func	Modified By Func
`_const`			`constructor`
`_secondsOfWeek`		`constructor`	
`_secondsOfTenYears`		`constructor`	
`validStakeStartTime`		`constructor`	
`_decimalLength`		`constructor`	
`_claimedTokens`		`constructor`	`unlockReward`

Review Comments

Review commit hash

- Review 1 [01/15/2020]: `4cfc290c94d8690618109e66ac57d5d9731f972f`
- Review 2 [01/19/2020]: `d1210c33a6e4565f4dc7edaf5a19c2cb6559944e`

AdminRole.sol

`AdminRole` follows inherits the openzeppelin's `WhitelistAdminRole` with added feature majorly focuses on having an array object `_adminList` to record admin addresses besides the mapping one. We assume two benefits based on the implementation: 1) add a prevention that the invocation of `remove()` would not erase the last admin which could lead to admin-less; 2) easier access to read all admins in an array than a mapping.

- INFO Recommend removing the below statement as it is never used in the contract. `using Roles for Roles.Role`
  - ✓ Hiblocks The code is removed and reflected in the latest commit

LockableToken.sol

`LockableToken` is a minimal-changed version based on `https://github.com/nitika-goel/lockable-token`.

- INFO `isAdminOrSelf()`: Typo on @dev note: is msg.sender or is whitelistadmin
- INFO Recommend ensuring `_to` is not a zero address.
  - ✓ Hiblocks The require check is added and reflected in the latest commit

StakingToken.sol

- DISCUSSION `constructor`: The constructor comment is describing as `Initializes the contract in stake state. Assigns the whitelistAdmin role to the deployer. .` However, the code is not reflecting as described.
  - ✓ Hiblocks The comment is updated and reflected in the latest commit
- INFO Recommend adding event logs for below critical functions.
  - `setvalidStakePeriod`
  - ✓ Hiblocks The event `validStakePeriod` is added and reflected in the latest commit
- INFO Recommend using external for following functions as gas optimization, given that the public function uses 496 gas, while the external function uses only 261.
  - `createStake`
  - `removeStake`
  - `stakeOf`
  - `stakeholderList`

  - ✓ Hiblocks The function visibility is updated and reflected in the latest commit

PausableToken.sol

- INFO `Versioning`: Inconsistent version, recommend using version `^0.5.6`
  - ✓ Hiblocks The version is updated and reflected in the latest commit

HiblocksIERC20.sol

- INFO `Versioning`: Inconsistent version,recommend using version `^0.5.6`
  - ✓ Hiblocks The version is updated and reflected in the latest commit

HiblockToken.sol

- INFO Recommend using external for follwoing functions as gas optimization, given that the public function uses 496 gas, while the external function uses only 261.
  - `memoOf`
  - `totalBalanceOf`
  - ✓ Hiblocks The function visibility is updated and reflected in the latest commit

RewardPool.sol

- INFO Recommend using external for following functions as gas optimization, given that the public function uses 496 gas, while the external function uses only 261.
  - `token`
  - `rewardOf`
  - ✓ Hiblocks The function visibility is updated and reflected in the latest commit
- MINOR `payReward`: Recommend using SafeMath library for mimic integer overflow issue.
  - ✓ Hiblocks The code is updated and reflected in the latest commit

```
function payReward(address recipient, uint256 amount, uint8 week, bytes memory
    memo) public onlyWhitelistAdmin returns (bool) {
    ...
    rewardToken[recipient][week] = rewardToken[recipient][week].add(amount);
    return true;
}
```

RewardSupplier.sol

- INFO `_factor()` is a one time setting function. From gas optimization, recommend to return a constant value `1300` with the explanation comments how the number derived.
  - ✓ Hiblocks The code is updated and reflected in the latest commit

```
/**
 * uint256 x1 = 1; // first week
 * uint256 y1 = 3000000; // reward tokens for first week
 * uint256 x2 = 52 * 2; // last week of 2 years
 * uint256 y2 = 16800000; // reward tokens for last week
 * factor = (y2.sub(y1)).div((x2.sub(x1)).mul((x2.sub(x1))));
 * @return factor value for calculate increased reward
 */
function _factor() internal pure returns (uint256 factor) {
    factor = 1300;
}
```

- MINOR Below functions are missing return value as defined in function signature:
  - `setBeneficiary`
  - `extendReleaseTime`
  - `withdrawalRemain`
  - ✓ Hiblocks The code is updated and reflected in the latest commit
- MINOR `tokensUnlockableUntil`: Recommend using SafeMath operation for avoiding the integer overflow
  - ✓ Hiblocks The code is updated and reflected in the latest commit

```
function tokensUnlockableUntil(uint256 week) {
...
total = total.add(tokensUnlockableAt(i));
...
}
```

- INFO Test Failure: The below unit test failed occasionally.

```
=== Test info ==============
totalStakes result
stakeValue : 1450
          ✓ get correct amount when try createStake several times (417ms)


  141 passing (47s)
  1 failing

  1) Contract: HiblocksToken
       ERC20::ERC1132
         extendLock
           when trying to extend Positive integer
             should extend lock time:

     AssertionError: expected '1' to equal '0'
     + expected - actual

     -1
     +0

     at Context.it (test/LockableToken.test.js:549:68)
     at process._tickCallback (internal/process/next_tick.js:68:7)
```

Best practice

Smart contract development requires a particular engineering mindset. A failure in the initial construction can be catastrophic, and changing the project after the fact can be exceedingly difficult.
To ensure success and to avoid the challenges above smart contracts should here to best practices at their conception. Below, we summarized a checklist of key points & vulnerability vectors that help to indicate a high overall quality of the current Hiblocks project. (✓ indicates satisfaction; × indicates unsatisfaction; − indicates inapplicablility)

General

Compiling

- ✓ Correct environment settings, e.g. compiler version, test framework
- ✓ No compiler warnings

Logging

- ✓ Provide error message along with `assert` & `require`
- ✓ Use events to monitor contract activities

Code Layout

- ✓ According to Solidity Tutorial, Layout contract elements should following below order:
  1. Pragma statements
  2. Import statements
  3. Interfaces
  4. Libraries

5. Contracts

✓ Each contract, library or interface should following below order:

1. Type declarations
2. State variables
3. Events
4. Functions

× According to Solidity Tutorial, functions should be grouped according to their visibility and ordered:

1. constructor
2. fallback function (if exists)
3. external
4. public
5. internal
6. private

Arithmetic Vulnerability

EVM specifies fixed-size data types for integers, in which means that has only a certain range of numbers it can store or represent.
Two's Complement / Integer underflow / overflow

✓ Use Math library as SafeMath for all arithmetic operations to handle integer overflow and underflow

Floating Points and Precision

✓ Correct handling the right precision when dealing ratios and rates

Access & Privilege Control Vulnerability

Circuit Breaker

✓ Provide pause functionality for control and emergency handling

Restriction

✓ Provide proper access control for functions

✓ Establish rate limiter for certain operations

✓ Restrict access to sensitive functions

✓ Restrict permission to contract destruction

✓ Establish speed bumps slow down some sensitive actions, any malicious actions occur, there is time to recover.

DoS Vulnerability

A type of attacks that make the contract inoperable with certain period of time or permanently.
Unexpected Revert

- ✓ Use favor pull over push pattern for handling unexpected revert

Block Gas Limit

- ✓ Use favor pull over push pattern for handling gas spent exceeds its limit on Contract via unbounded operations
- ✓ Use favor pull over push pattern for handling gas spent exceeds its limit on the network via block stuffing

Miner Manipulation Vulnerability

BlockNumber Dependence

- ✓ Understand the security risk level and trade-off of using `block.number` as one of core factors in the contract. Be aware that `block.number` can not be manipulated by the miner, but can lead to large than expected time differences. With assumptions of an Ethereum block confirmation takes 13 seconds. However, the average block time is between 13 15 seconds. During the difficulty bomb stage or hard/soft fork upgrade of the network, `block.number` to a time is dangerous and inaccurate as expected.

Timestamp Dependence

- – Understand the security risk level and trade-off of using `block.timestamp` or alias `now` as one of core factors in the contract.
- – Correct use of 15-second rule to minimize the impact caused by timestamp variance

Transaction Ordering Or Front-Running

- – Understand the security risk level and the `gasPrice` rule in this vulnerability
- – Correct placing an upper bound on the `gasPrice` for preventing the users taking the benefit of transaction ordering

External Referencing Vulnerability

External calls may execute malicious code in that contract or any other contract that it depends upon. As such, every external call should be treated as a potential security risk

- × Correct using the pull over push favor for external calls to reduce reduces the chance of problems with the gas limit.

Avoid state changes after external calls

- × Correct using checks-effects-interactions pattern to minimize the state changes after external contract or call referencing.

Handle errors in external calls

- ✓ Correct handling errors in any external contract or call referencing by checking its return value

Race Conditions Vulnerability

A type of vulnerability caused by calling external contracts that attacker can take over the control flow, and make changes to the data that the calling function wasn't expecting.

- Type of race conditions:
  - Reentrancy
    A state variable is changed after a contract uses `call.value()()`.
  - Cross-function Race Conditions

```
\begin{lstlisting}[language=Solidity,numbers=none]
An attacker may also be able to do a similar attack using two different functi
\end{lstlisting}
```

✓ Avoid using `call.value()()`, instead use `send()`, `transfer()` that consumes 2300 gas. This will prevent any external code from being executed continuously

× Finish all internal work before calling the external function for unavoidable external call.

Low-level Call Vulnerability

The low-level function or opcodes are very useful and danger as for allowing the Libraries implementation and modularized code. However it opens up the doors to vulnerabilities as essentially your contract is allowing anyone to do whatever they want with their state
Code Injection by delegatecall

✓ Ensure the libraries implementation is stateless and non-self-destructable

Visibility Vulnerability

Solidity functions have 4 difference visibility dictate how functions are allowed to be called. The visibility determines whether a function can be called externally by users, by other derived contracts, only internally or only externally.

✓ Specify the visibility of all functions in a contract, even if they are intentionally public

Incorrect Interface Vulnerability

A contract interface defines functions with a different type signature than the implementation, causing two different method id's to be created. As a result, when the interface is called, the fallback method will be executed.

✓ Ensure the defined function signatures are match with the contract interface and implementation

Bad Randomness Vulnerability

Pseudo random number generation is not supported by Solidity as default, which it is an unsafe operation.

✓ Avoid using randomness for block variables, there may be a chance manipulated by the miners

Documentation

- ✓ Provide project README and execution guidance
- ✓ Provide inline comment for complex functions intention
- ✓ Provide instruction to initialize and execute the test files

Testing

- ✓ Provide migration scripts for easy contracts deployment to the Ethereum network
- ✓ Provide test scripts and coverage for potential scenarios

Overall we found the smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend to seek multiple opinions, keep improving the codebase, and more test coverage and sandbox deployments before the mainnet release.

Static Analysis Results

TIMESTAMP_DEPENDENCY

Line 201 in File RewardSupplier.sol

```
201         require(_const.releaseTime < now, "not a release period."); //solium-disable-line
```

! "now" can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 211 in File RewardSupplier.sol

```
211         require(_const.releaseTime < now, "not a release period."); //solium-disable-line
```

! "now" can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 221 in File RewardSupplier.sol

```
221             _claimedTokens.push(ClaimedToken(reward, week, now)); //solium-disable-line
```

! "now" can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 260 in File RewardSupplier.sol

```
260         require(_const.releaseTime > now, "release time is in past"); //solium-disable-line
```

! "now" can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 275 in File RewardSupplier.sol

```
275         require(endTime() < now, "Reward period is left."); //solium-disable-line
```

! "now" can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 114 in File StakingToken.sol

```
114     if(!stakeFlag || now < validStakeStartTime) { //solium-disable-line
```

! "now" can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 125 in File StakingToken.sol

```
125     if(now < validStakeStartTime) { //solium-disable-line
```

! "now" can be influenced by miners to some degree

TIMESTAMP_DEPENDENCY

Line 143 in File LockableToken.sol

```
143     if (locked[_of][_reason].validity <= now && !locked[_of][_reason].claimed) //solium-
            disable-line
```

! "now" can be influenced by miners to some degree

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms

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

CERTIK *label*

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

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

Raw code

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

Counterexample: This code violates the specification

Initial environment

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          from = 0x0
5          to = 0x0
6          tokens = 0x6c
7      }
8      This = 0
                    }
53              balance: 0x0
54          }
55      }
56
```

Post environment

```
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

Formal Verification Request 1

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

20, Jan 2020
6.72 ms

Line 77 in File RewardSupplier.sol

```
77    //@CTK NO_BUF_OVERFLOW
```

Line 80-88 in File RewardSupplier.sol

```
80    function _factor() internal pure returns (uint256 factor) {
81        // uint256 x1 = 1; // first week
82        // uint256 y1 = 3000000; // reward tokens for first week
83        // uint256 x2 = 52 * 2; // last week of 2 years
84        // uint256 y2 = 16800000; // reward tokens for last week
85
86        // factor = (y2.sub(y1)).div((x2.sub(x1)).mul((x2.sub(x1))));
87        factor = 1300;
88    }
```

The code meets the specification.

Formal Verification Request 2

If method completes, integer overflow would not happen.

20, Jan 2020
1.28 ms

Line 78 in File RewardSupplier.sol

```
78    //@CTK NO_OVERFLOW
```

Line 80-88 in File RewardSupplier.sol

```
80    function _factor() internal pure returns (uint256 factor) {
81        // uint256 x1 = 1; // first week
82        // uint256 y1 = 3000000; // reward tokens for first week
83        // uint256 x2 = 52 * 2; // last week of 2 years
84        // uint256 y2 = 16800000; // reward tokens for last week
85
86        // factor = (y2.sub(y1)).div((x2.sub(x1)).mul((x2.sub(x1))));
87        factor = 1300;
88    }
```

The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.

20, Jan 2020
0.81 ms

Line 79 in File RewardSupplier.sol

```
79    //@CTK NO_ASF
```

Line 80-88 in File RewardSupplier.sol

```
80    function _factor() internal pure returns (uint256 factor) {
81        // uint256 x1 = 1; // first week
82        // uint256 y1 = 3000000; // reward tokens for first week
83        // uint256 x2 = 52 * 2; // last week of 2 years
84        // uint256 y2 = 16800000; // reward tokens for last week
85
86        // factor = (y2.sub(y1)).div((x2.sub(x1)).mul((x2.sub(x1))));
87        factor = 1300;
88    }
```

✅ The code meets the specification.

Formal Verification Request 4

token

📅 20, Jan 2020
⏱ 11.22 ms

Line 93-95 in File RewardSupplier.sol

```
93    /*@CTK token
94      @post __return == _token
95    */
```

Line 96-98 in File RewardSupplier.sol

```
96    function token() external view returns (HiblocksIERC20) {
97        return _token;
98    }
```

✅ The code meets the specification.

Formal Verification Request 5

releaseTime

📅 20, Jan 2020
⏱ 14.68 ms

Line 103-105 in File RewardSupplier.sol

```
103   /*@CTK releaseTime
104     @post __return == _const.releaseTime
105   */
```

Line 106-108 in File RewardSupplier.sol

```
106   function releaseTime() external view returns (uint256) {
107       return _const.releaseTime;
108   }
```

✅ The code meets the specification.

Formal Verification Request 6

beneficiary

20, Jan 2020
16.95 ms

Line 113-115 in File RewardSupplier.sol

```
113   /*@CTK beneficiary
114     @post __return == _const.beneficiary
115   */
```

Line 116-118 in File RewardSupplier.sol

```
116   function beneficiary() external view returns (address) {
117       return _const.beneficiary;
118   }
```

The code meets the specification.

Formal Verification Request 7

beneficiary

20, Jan 2020
134.0 ms

Line 123-125 in File RewardSupplier.sol

```
123   /*@CTK beneficiary
124     @post __return == _const.totalReward * (10**18)
125   */
```

Line 126-128 in File RewardSupplier.sol

```
126   function totalReward() external view returns (uint256) {
127       return (_const.totalReward.mul(10**18));
128   }
```

The code meets the specification.

Formal Verification Request 8

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

20, Jan 2020
132.16 ms

Line 140 in File RewardSupplier.sol

```
140   //@CTK NO_BUF_OVERFLOW
```

Line 147-149 in File RewardSupplier.sol

```
147   function endTime() public view returns (uint256) {
148       return _const.releaseTime.add(_secondsOfTenYears); // end time of reward
              distribution;
149   }
```

The code meets the specification.

Formal Verification Request 9

If method completes, integer overflow would not happen.

20, Jan 2020
9.38 ms

Line 141 in File RewardSupplier.sol

```
141     //@CTK NO_OVERFLOW
```

Line 147-149 in File RewardSupplier.sol

```
147     function endTime() public view returns (uint256) {
148         return _const.releaseTime.add(_secondsOfTenYears); // end time of reward
                distribution;
149     }
```

The code meets the specification.

Formal Verification Request 10

Method will not encounter an assertion failure.

20, Jan 2020
2.16 ms

Line 142 in File RewardSupplier.sol

```
142     //@CTK NO_ASF
```

Line 147-149 in File RewardSupplier.sol

```
147     function endTime() public view returns (uint256) {
148         return _const.releaseTime.add(_secondsOfTenYears); // end time of reward
                distribution;
149     }
```

The code meets the specification.

Formal Verification Request 11

endTime

20, Jan 2020
12.54 ms

Line 143-146 in File RewardSupplier.sol

```
143     /*@CTK "endTime"
144       @tag assume_completion
145       @post __return == _const.releaseTime + _secondsOfTenYears
146     */
```

Line 147-149 in File RewardSupplier.sol

```
147     function endTime() public view returns (uint256) {
148         return _const.releaseTime.add(_secondsOfTenYears); // end time of reward
                distribution;
149     }
```

The code meets the specification.

Formal Verification Request 12

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

20, Jan 2020
726.38 ms

Line 192 in File RewardSupplier.sol

```
192     //@CTK NO_BUF_OVERFLOW
```

Line 200-203 in File RewardSupplier.sol

```
200     function thisWeek() public view returns (uint256){
201         require(_const.releaseTime < now, "not a release period."); //solium-disable-line
202         return now.sub(_const.releaseTime).div(_secondsOfWeek); //solium-disable-line
203     }
```

The code meets the specification.

Formal Verification Request 13

If method completes, integer overflow would not happen.

20, Jan 2020
18.91 ms

Line 193 in File RewardSupplier.sol

```
193     //@CTK NO_OVERFLOW
```

Line 200-203 in File RewardSupplier.sol

```
200     function thisWeek() public view returns (uint256){
201         require(_const.releaseTime < now, "not a release period."); //solium-disable-line
202         return now.sub(_const.releaseTime).div(_secondsOfWeek); //solium-disable-line
203     }
```

The code meets the specification.

Formal Verification Request 14

Method will not encounter an assertion failure.

20, Jan 2020
10.83 ms

Line 194 in File RewardSupplier.sol

```
194    //@CTK NO_ASF
```

Line 200-203 in File RewardSupplier.sol

```
200    function thisWeek() public view returns (uint256){
201        require(_const.releaseTime < now, "not a release period."); //solium-disable-line
202        return now.sub(_const.releaseTime).div(_secondsOfWeek); //solium-disable-line
203    }
```

The code meets the specification.

Formal Verification Request 15

thisWeek

20, Jan 2020
1389.49 ms

Line 195-199 in File RewardSupplier.sol

```
195    /*@CTK "thisWeek"
196      @tag assume_completion
197      @post _const.releaseTime >= now -> __reverted
198      @post __return == (now - _const.releaseTime) / _secondsOfWeek
199    */
```

Line 200-203 in File RewardSupplier.sol

```
200    function thisWeek() public view returns (uint256){
201        require(_const.releaseTime < now, "not a release period."); //solium-disable-line
202        return now.sub(_const.releaseTime).div(_secondsOfWeek); //solium-disable-line
203    }
```

The code meets the specification.

Formal Verification Request 16

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

20, Jan 2020
10.99 ms

Line 235 in File RewardSupplier.sol

```
235    //@CTK NO_BUF_OVERFLOW
```

Line 241-245 in File RewardSupplier.sol

```
241    function setBeneficiary(address newBeneficiary) external onlyWhitelistAdmin returns (
           bool) {
242        _const.beneficiary = newBeneficiary;
243        emit RewardLocked(newBeneficiary, _const.releaseTime);
244        return true;
245    }
```

The code meets the specification.

Formal Verification Request 17

If method completes, integer overflow would not happen.

20, Jan 2020
0.43 ms

Line 236 in File RewardSupplier.sol

```
236     //@CTK NO_OVERFLOW
```

Line 241-245 in File RewardSupplier.sol

```
241     function setBeneficiary(address newBeneficiary) external onlyWhitelistAdmin returns (
            bool) {
242         _const.beneficiary = newBeneficiary;
243         emit RewardLocked(newBeneficiary, _const.releaseTime);
244         return true;
245     }
```

The code meets the specification.

Formal Verification Request 18

Method will not encounter an assertion failure.

20, Jan 2020
0.4 ms

Line 237 in File RewardSupplier.sol

```
237     //@CTK NO_ASF
```

Line 241-245 in File RewardSupplier.sol

```
241     function setBeneficiary(address newBeneficiary) external onlyWhitelistAdmin returns (
            bool) {
242         _const.beneficiary = newBeneficiary;
243         emit RewardLocked(newBeneficiary, _const.releaseTime);
244         return true;
245     }
```

The code meets the specification.

Formal Verification Request 19

setBeneficiary

20, Jan 2020
0.44 ms

Line 238-240 in File RewardSupplier.sol

```
238     /*@CTK "setBeneficiary"
239       @post _const.beneficiary == newBeneficiary -> __return == true
240     */
```

Line 241-245 in File RewardSupplier.sol

```
241     function setBeneficiary(address newBeneficiary) external onlyWhitelistAdmin returns (
            bool) {
242         _const.beneficiary = newBeneficiary;
243         emit RewardLocked(newBeneficiary, _const.releaseTime);
244         return true;
245     }
```

The code meets the specification.

Formal Verification Request 20

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

20, Jan 2020
55.86 ms

Line 251 in File RewardSupplier.sol

```
251     //@CTK NO_BUF_OVERFLOW
```

Line 259-265 in File RewardSupplier.sol

```
259     function extendReleaseTime(uint256 extendTime) external onlyWhitelistAdmin returns (bool
            ) {
260         require(_const.releaseTime > now, "release time is in past"); //solium-disable-line
261 
262         _const.releaseTime = _const.releaseTime.add(extendTime);
263         emit RewardLocked(_const.beneficiary, _const.releaseTime);
264         return true;
265     }
```

The code meets the specification.

Formal Verification Request 21

If method completes, integer overflow would not happen.

20, Jan 2020
4.76 ms

Line 252 in File RewardSupplier.sol

```
252     //@CTK NO_OVERFLOW
```

Line 259-265 in File RewardSupplier.sol

```
259     function extendReleaseTime(uint256 extendTime) external onlyWhitelistAdmin returns (bool
            ) {
260         require(_const.releaseTime > now, "release time is in past"); //solium-disable-line
261 
262         _const.releaseTime = _const.releaseTime.add(extendTime);
263         emit RewardLocked(_const.beneficiary, _const.releaseTime);
264         return true;
265     }
```

The code meets the specification.

Formal Verification Request 22

Method will not encounter an assertion failure.

20, Jan 2020
0.92 ms

Line 253 in File RewardSupplier.sol

```
253     //@CTK NO_ASF
```

Line 259-265 in File RewardSupplier.sol

```
259     function extendReleaseTime(uint256 extendTime) external onlyWhitelistAdmin returns (bool
            ) {
260         require(_const.releaseTime > now, "release time is in past"); //solium-disable-line
261 
262         _const.releaseTime = _const.releaseTime.add(extendTime);
263         emit RewardLocked(_const.beneficiary, _const.releaseTime);
264         return true;
265     }
```

The code meets the specification.

Formal Verification Request 23

constructor

20, Jan 2020
662.74 ms

Line 17-19 in File HiblocksToken.sol

```
17  /*@CTK "constructor"
18    @post initialSupply <= 0 -> __reverted
19  */
```

Line 20-27 in File HiblocksToken.sol

```
20  constructor(string memory name, string memory symbol, uint8 decimals, uint256
        initialSupply)
21    public
22    ERC20Detailed(name, symbol, decimals)
23  {
24    // Mint the initial supply
25    require(initialSupply > 0, "initialSupply must be greater than zero.");
26    _mint(_msgSender(), initialSupply * (10 ** uint256(decimals)));
27  }
```

The code meets the specification.

Formal Verification Request 24

transferWithMemo

20, Jan 2020
116.92 ms

Line 35-39 in File HiblocksToken.sol

```
35  /*@CTK "transferWithMemo"
36    @tag assume_completion
37    @post __post.memos[to][block.number] == memo
38    @post __return == true
39  */
```

Line 40-46 in File HiblocksToken.sol

```
40  function transferWithMemo(address to, uint256 value, bytes memory memo) public returns (
        bool) {
41    require(transfer(to, value), "Token transfer failed");
42    memos[to][block.number] = memo;
43    return true;
44  }
```

The code meets the specification.

Formal Verification Request 25

memoOf

20, Jan 2020
13.91 ms

Line 54-56 in File HiblocksToken.sol

```
54  /*@CTK "memoOf"
55    @post __return == memos[addr][blockNumber]
56  */
```

Line 57-59 in File HiblocksToken.sol

```
57  function memoOf(address addr, uint blockNumber) external view returns (bytes memory) {
58    return(memos[addr][blockNumber]);
59  }
```

The code meets the specification.

Formal Verification Request 26

Method will not encounter an assertion failure.

20, Jan 2020
9.97 ms

Line 28 in File RewardPool.sol

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

Line 29-31 in File RewardPool.sol

```
29    constructor(address tokenAddr) public {
30        _token = HiblocksIERC20(tokenAddr);
31    }
```

The code meets the specification.

Formal Verification Request 27

token

20, Jan 2020
7.89 ms

Line 36-38 in File RewardPool.sol

```
36    /*@CTK "token"
37      @post __return == _token
38    */
```

Line 39-41 in File RewardPool.sol

```
39    function token() external view returns (HiblocksIERC20) {
40        return _token;
41    }
```

The code meets the specification.

Formal Verification Request 28

rewardOf

20, Jan 2020
9.98 ms

Line 55-57 in File RewardPool.sol

```
55    /*@CTK "rewardOf"
56      @post __return == rewardToken[recipient][week]
57    */
```

Line 58-60 in File RewardPool.sol

```
58    function rewardOf(address recipient, uint8 week) external view returns (uint256) {
59        return rewardToken[recipient][week];
60    }
```

The code meets the specification.

Formal Verification Request 29

constructor

20, Jan 2020
17.26 ms

Line 38-40 in File PausableToken.sol

```
38  /*@CTK "constructor"
39    @post _paused == false
40  */
```

Line 41-43 in File PausableToken.sol

```
41  constructor () internal {
42    _paused = false;
43  }
```

The code meets the specification.

Formal Verification Request 30

paused

20, Jan 2020
13.63 ms

Line 48-50 in File PausableToken.sol

```
48  /*@CTK "paused"
49    @post __return == _paused
50  */
```

Line 51-53 in File PausableToken.sol

```
51  function paused() public view returns (bool) {
52    return _paused;
53  }
```

The code meets the specification.

Formal Verification Request 31

pause

20, Jan 2020
277.23 ms

Line 74-80 in File PausableToken.sol

```
74  /*@CTK pause
75    @post _paused -> __reverted
76    @post msg.sender == address(0) -> __reverted
77    @post !_whitelistAdmins.bearer[msg.sender] -> __reverted
78    @post !_paused && msg.sender != address(0) && _whitelistAdmins.bearer[msg.sender] &&
          _pausers.bearer[msg.sender]
79        -> __post._paused
80  */
```

Line 81-84 in File PausableToken.sol

```
81  function pause() public onlyWhitelistAdmin whenNotPaused {
82    _paused = true;
83    emit Paused(_msgSender());
84  }
```

The code meets the specification.

Formal Verification Request 32

unpause

20, Jan 2020
124.0 ms

Line 89-95 in File PausableToken.sol

```
89  /*@CTK unpause
90    @post !_paused -> __reverted
91    @post msg.sender == address(0) -> __reverted
92    @post !_whitelistAdmins.bearer[msg.sender] -> __reverted
93    @post _paused && msg.sender != address(0) && _whitelistAdmins.bearer[msg.sender] &&
          _pausers.bearer[msg.sender]
94        -> !__post._paused
95  */
```

Line 96-99 in File PausableToken.sol

```
96  function unpause() public onlyWhitelistAdmin whenPaused {
97    _paused = false;
98    emit Unpaused(_msgSender());
99  }
```

The code meets the specification.

Formal Verification Request 33

If method completes, integer overflow would not happen.

20, Jan 2020
724.58 ms

Line 107 in File PausableToken.sol

```
107  //@CTK NO_OVERFLOW
```

Line 118-120 in File PausableToken.sol

```
118  function transfer(address to, uint256 value) public whenNotPaused returns (bool) {
119    return super.transfer(to, value);
120  }
```

The code meets the specification.

Formal Verification Request 34

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

20, Jan 2020
55.72 ms

Line 108 in File PausableToken.sol

```
108  //@CTK NO_BUF_OVERFLOW
```

Line 118-120 in File PausableToken.sol

```
118  function transfer(address to, uint256 value) public whenNotPaused returns (bool) {
119    return super.transfer(to, value);
120  }
```

The code meets the specification.

Formal Verification Request 35

Method will not encounter an assertion failure.

20, Jan 2020
52.11 ms

Line 109 in File PausableToken.sol

```
109  //@CTK NO_ASF
```

Line 118-120 in File PausableToken.sol

```
118  function transfer(address to, uint256 value) public whenNotPaused returns (bool) {
119    return super.transfer(to, value);
120  }
```

The code meets the specification.

Formal Verification Request 36

transfer

20, Jan 2020
415.7 ms

Line 110-117 in File PausableToken.sol

```
110  /*@CTK "transfer"
111    @tag assume_completion
112    @post (to == address(0) || _paused == true) -> __reverted
113    @post value <= _balances[msg.sender]
114    @post (to != address(0) && to != msg.sender) -> __post._balances[msg.sender] ==
           _balances[msg.sender] - value
115    @post (to != address(0) && to != msg.sender) -> __post._balances[to] == _balances[to] +
           value
116    @post (to != address(0) && to == msg.sender) -> __post._balances[msg.sender] ==
           _balances[msg.sender]
117  */
```

Line 118-120 in File PausableToken.sol

```
118  function transfer(address to, uint256 value) public whenNotPaused returns (bool) {
119    return super.transfer(to, value);
120  }
```

The code meets the specification.

Formal Verification Request 37

If method completes, integer overflow would not happen.

20, Jan 2020
671.53 ms

Line 129 in File PausableToken.sol

```
129  //@CTK NO_OVERFLOW
```

Line 141-143 in File PausableToken.sol

```
141  function transferFrom(address from, address to, uint256 value) public whenNotPaused
         returns (bool) {
142    return super.transferFrom(from, to, value);
143  }
```

The code meets the specification.

Formal Verification Request 38

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

20, Jan 2020
76.91 ms

Line 130 in File PausableToken.sol

```
130  //@CTK NO_BUF_OVERFLOW
```

Line 141-143 in File PausableToken.sol

```
141  function transferFrom(address from, address to, uint256 value) public whenNotPaused
         returns (bool) {
142    return super.transferFrom(from, to, value);
143  }
```

The code meets the specification.

Formal Verification Request 39

Method will not encounter an assertion failure.

20, Jan 2020
74.26 ms

Line 131 in File PausableToken.sol

```
131  //@CTK NO_ASF
```

Line 141-143 in File PausableToken.sol

```
141  function transferFrom(address from, address to, uint256 value) public whenNotPaused
         returns (bool) {
142    return super.transferFrom(from, to, value);
143  }
```

The code meets the specification.

Formal Verification Request 40

transferFrom

20, Jan 2020
721.01 ms

Line 132-140 in File PausableToken.sol

```
132  /*@CTK "transferFrom"
133    @tag assume_completion
134    @post (_paused || msg.sender == address(0) || from == address(0) || to == address(0)) ->
           __reverted
135    @post from == to -> __post._balances[from] == _balances[from]
136    @post from != to -> __post._balances[from] == _balances[from] - value
137    @post from != to -> __post._balances[to] == _balances[to] + value
138    @post __post._allowances[from][msg.sender] == _allowances[from][msg.sender] - value
139    @post __return == true
140    */
```

Line 141-143 in File PausableToken.sol

```
141  function transferFrom(address from, address to, uint256 value) public whenNotPaused
         returns (bool) {
142    return super.transferFrom(from, to, value);
143  }
```

The code meets the specification.

Formal Verification Request 41

If method completes, integer overflow would not happen.

20, Jan 2020
147.21 ms

Line 151 in File PausableToken.sol

```
151  //@CTK NO_OVERFLOW
```

Line 161-163 in File PausableToken.sol

```
161  function approve(address spender, uint256 value) public whenNotPaused returns (bool) {
162    return super.approve(spender, value);
163  }
```

The code meets the specification.

Formal Verification Request 42

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

20, Jan 2020
1.34 ms

Line 152 in File PausableToken.sol

```
152  //@CTK NO_BUF_OVERFLOW
```

Line 161-163 in File PausableToken.sol

```
161  function approve(address spender, uint256 value) public whenNotPaused returns (bool) {
162    return super.approve(spender, value);
163  }
```

✅ The code meets the specification.

Formal Verification Request 43

Method will not encounter an assertion failure.

📅 20, Jan 2020
⏱ 1.13 ms

Line 153 in File PausableToken.sol

```
153  //@CTK NO_ASF
```

Line 161-163 in File PausableToken.sol

```
161  function approve(address spender, uint256 value) public whenNotPaused returns (bool) {
162    return super.approve(spender, value);
163  }
```

✅ The code meets the specification.

Formal Verification Request 44

approve

📅 20, Jan 2020
⏱ 21.98 ms

Line 154-160 in File PausableToken.sol

```
154  /*@CTK "approve"
155    @tag assume_completion
156    @post (_paused || msg.sender == address(0) || spender == address(0)) -> __reverted
157    @post (!_paused || msg.sender != address(0) || spender != address(0)) ->
158      __post._allowances[msg.sender][spender] == value
159    @post __return == true
160    */
```

Line 161-163 in File PausableToken.sol

```
161  function approve(address spender, uint256 value) public whenNotPaused returns (bool) {
162    return super.approve(spender, value);
163  }
```

✅ The code meets the specification.

Formal Verification Request 45

If method completes, integer overflow would not happen.

📅 20, Jan 2020
⏱ 331.79 ms

Line 172 in File PausableToken.sol

```
172  //@CTK NO_OVERFLOW
```

Line 181-183 in File PausableToken.sol

```
181  function increaseAllowance(address spender, uint256 addedValue) public whenNotPaused
         returns (bool) {
182    return super.increaseAllowance(spender, addedValue);
183  }
```

✅ The code meets the specification.

Formal Verification Request 46

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

📅 20, Jan 2020
⏱ 3.4 ms

Line 173 in File PausableToken.sol

```
173  //@CTK NO_BUF_OVERFLOW
```

Line 181-183 in File PausableToken.sol

```
181  function increaseAllowance(address spender, uint256 addedValue) public whenNotPaused
         returns (bool) {
182    return super.increaseAllowance(spender, addedValue);
183  }
```

✅ The code meets the specification.

Formal Verification Request 47

Method will not encounter an assertion failure.

📅 20, Jan 2020
⏱ 3.44 ms

Line 174 in File PausableToken.sol

```
174  //@CTK NO_ASF
```

Line 181-183 in File PausableToken.sol

```
181  function increaseAllowance(address spender, uint256 addedValue) public whenNotPaused
         returns (bool) {
182    return super.increaseAllowance(spender, addedValue);
183  }
```

✅ The code meets the specification.

Formal Verification Request 48

increaseAllowance

20, Jan 2020
100.24 ms

Line 175-180 in File PausableToken.sol

```
175  /*@CTK "increaseAllowance"
176    @tag assume_completion
177    @post (_paused == true || spender == address(0)) -> __reverted
178    @post (_paused == false && spender != address(0)) -> __post._allowances[msg.sender][
           spender] == _allowances[msg.sender][spender] + addedValue
179    @post __return == true
180  */
```

Line 181-183 in File PausableToken.sol

```
181  function increaseAllowance(address spender, uint256 addedValue) public whenNotPaused
         returns (bool) {
182    return super.increaseAllowance(spender, addedValue);
183  }
```

The code meets the specification.

Formal Verification Request 49

If method completes, integer overflow would not happen.

20, Jan 2020
338.84 ms

Line 191 in File PausableToken.sol

```
191  //@CTK NO_OVERFLOW
```

Line 199-201 in File PausableToken.sol

```
199  function decreaseAllowance(address spender, uint256 subtractedValue) public whenNotPaused
         returns (bool) {
200    return super.decreaseAllowance(spender, subtractedValue);
201  }
```

The code meets the specification.

Formal Verification Request 50

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

20, Jan 2020
3.8 ms

Line 192 in File PausableToken.sol

```
192  //@CTK NO_BUF_OVERFLOW
```

Line 199-201 in File PausableToken.sol

```
199  function decreaseAllowance(address spender, uint256 subtractedValue) public whenNotPaused
         returns (bool) {
200    return super.decreaseAllowance(spender, subtractedValue);
201  }
```

The code meets the specification.

Formal Verification Request 51

Method will not encounter an assertion failure.

20, Jan 2020
4.2 ms

Line 193 in File PausableToken.sol

```
193  //@CTK NO_ASF
```

Line 199-201 in File PausableToken.sol

```
199  function decreaseAllowance(address spender, uint256 subtractedValue) public whenNotPaused
         returns (bool) {
200    return super.decreaseAllowance(spender, subtractedValue);
201  }
```

The code meets the specification.

Formal Verification Request 52

decreaseAllowance

20, Jan 2020
123.3 ms

Line 194-198 in File PausableToken.sol

```
194  /*@CTK "decreaseAllowance"
195    @tag assume_completion
196    @post (_paused == true || spender == address(0)) -> __reverted
197    @post (_paused == false && spender != address(0)) -> __post._allowances[msg.sender][
           spender] == _allowances[msg.sender][spender] - subtractedValue
198    */
```

Line 199-201 in File PausableToken.sol

```
199  function decreaseAllowance(address spender, uint256 subtractedValue) public whenNotPaused
         returns (bool) {
200    return super.decreaseAllowance(spender, subtractedValue);
201  }
```

The code meets the specification.

Formal Verification Request 53

constructor

20, Jan 2020
40.23 ms

Line 43-48 in File StakingToken.sol

```
43  /*@CTK "constructor"
44    @post __post.secondsInterval == 604800
45    @post __post.validStakeDuration == 32400
46    @post __post.validStakeStartTime == now - 32400
47    @post __post.stakeFlag == false
48  */
```

Line 49-54 in File StakingToken.sol

```
49  constructor () internal {
50    secondsInterval = 60*60*24*7; // default : interval 7 days
51    validStakeDuration = 32400; // default : 60*60*9 ( 9hours )
52    validStakeStartTime = now - validStakeDuration; //solium-disable-line
53    stakeFlag = false;
54  }
```

The code meets the specification.

Formal Verification Request 54

isAbleToStake false

20, Jan 2020
232.38 ms

Line 100-103 in File StakingToken.sol

```
100 /*@CTK "isAbleToStake false"
101   @post (!stakeFlag || now < validStakeStartTime) -> __return == false
102
103 */
```

Line 113-119 in File StakingToken.sol

```
113 function isAbleToStake() public view returns (bool){
114   if(!stakeFlag || now < validStakeStartTime) { //solium-disable-line
115     return false;
116   }
117   uint256 validTime = now.sub(validStakeStartTime).mod(secondsInterval); //solium-disable-
          line
118   return (validTime >= 0 && validTime < validStakeDuration);
119 }
```

The code meets the specification.

Formal Verification Request 55

isAbleToStake

20, Jan 2020
9272.0 ms

Line 104-112 in File StakingToken.sol

```
104  /*@CTK FAIL "isAbleToStake "
105    @pre stakeFlag == true
106    @pre now >= validStakeStartTime
107    @pre now == validStakeStartTime
108    @pre now == secondsInterval
109    @pre now - validStakeStartTime % secondsInterval < validStakeDuration
110    @post __return == false
111
112  */
```

Line 113-119 in File StakingToken.sol

```
113  function isAbleToStake() public view returns (bool){
114    if(!stakeFlag || now < validStakeStartTime) { //solium-disable-line
115      return false;
116    }
117    uint256 validTime = now.sub(validStakeStartTime).mod(secondsInterval); //solium-disable-
           line
118    return (validTime >= 0 && validTime < validStakeDuration);
119  }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3     This = 0
4     Internal = {
5         __has_assertion_failure = false
6         __has_buf_overflow = false
7         __has_overflow = false
8         __has_returned = false
9         __reverted = false
10        msg = {
11          "gas": 0,
12          "sender": 0,
13          "value": 0
14        }
15    }
16    Other = {
17        ALREADY_LOCKED = "\u0095\u00b0\u00ac\u00a6\u00af\u00b4a\u00a2\u00ad\u00b3\u00a6\u00a
              2\u00a5\u00baa\u00ad\u00b0\u00a4\u00ac\u00a6\u00a5"
18        AMOUNT_ZERO = "\u0082\u00ae\u00b0\u00b6\u00af\u00b5a\u00a4\u00a2\u00afa\u00af\u00b0\
              u00b5a\u00a3\u00a6aq"
19        NOT_LOCKED = "\u008f\u00b0a\u00b5\u00b0\u00ac\u00a6\u00af\u00b4a\u00ad\u00b0\u00a4\u
              00ac\u00a6\u00a5"
20        __return = false
21        block = {
22          "number": 0,
23          "timestamp": 1
24        }
25    }
```

```
Address_Map = [
  {
    "key": 0,
    "value": {
      "contract_name": "StakingToken",
      "balance": 0,
      "contract": {
        "stakeholders": [],
        "stakes": [
          {
            "key": 0,
            "value": 16
          },
          {
            "key": 32,
            "value": 1
          },
          {
            "key": 8,
            "value": 2
          },
          {
            "key": "ALL_OTHERS",
            "value": 5
          }
        ],
        "secondsInterval": 1,
        "validStakeDuration": 2,
        "validStakeStartTime": 1,
        "stakeFlag": true,
        "_whitelistAdmins": {
          "bearer": [
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ]
        },
        "lockReason": [
          {
            "key": "ALL_OTHERS",
            "value": [
              "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
              "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
              "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
              "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
              "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
            ]
          }
        ],
        "locked": [
          {
            "key": "ALL_OTHERS",
            "value": [
```

```
                {
                  "key": "ALL_OTHERS",
                  "value": {
                    "amount": 5,
                    "validity": 5,
                    "claimed": false
                  }
                }
              ]
            }
          ],
          "_paused": false,
          "_balances": [
            {
              "key": 2,
              "value": 68
            },
            {
              "key": 0,
              "value": 4
            },
            {
              "key": "ALL_OTHERS",
              "value": 5
            }
          ],
          "_pausers": {
            "bearer": [
              {
                "key": "ALL_OTHERS",
                "value": false
              }
            ]
          },
          "_adminList": [],
          "_allowances": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 5
                }
              ]
            }
          ],
          "_totalSupply": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  This = 0
```

```
142     Internal = {
143         __has_assertion_failure = false
144         __has_buf_overflow = false
145         __has_overflow = false
146         __has_returned = true
147         __reverted = false
148         msg = {
149           "gas": 0,
150           "sender": 0,
151           "value": 0
152         }
153     }
154     Other = {
155         ALREADY_LOCKED = "\u0095\u00b0\u00ac\u00a6\u00af\u00b4a\u00a2\u00ad\u00b3\u00a6\u00a
            2\u00a5\u00baa\u00ad\u00b0\u00a4\u00ac\u00a6\u00a5"
156         AMOUNT_ZERO = "\u0082\u00ae\u00b0\u00b6\u00af\u00b5a\u00a4\u00a2\u00afa\u00af\u00b0\
            u00b5a\u00a3\u00a6aq"
157         NOT_LOCKED = "\u008f\u00b0a\u00b5\u00b0\u00ac\u00a6\u00af\u00b4a\u00ad\u00b0\u00a4\u
            00ac\u00a6\u00a5"
158         __return = true
159         block = {
160           "number": 0,
161           "timestamp": 1
162         }
163     }
164     Address_Map = [
165       {
166         "key": 0,
167         "value": {
168           "contract_name": "StakingToken",
169           "balance": 0,
170           "contract": {
171             "stakeholders": [],
172             "stakes": [
173               {
174                 "key": 0,
175                 "value": 16
176               },
177               {
178                 "key": 32,
179                 "value": 1
180               },
181               {
182                 "key": 8,
183                 "value": 2
184               },
185               {
186                 "key": "ALL_OTHERS",
187                 "value": 5
188               }
189             ],
190             "secondsInterval": 1,
191             "validStakeDuration": 2,
192             "validStakeStartTime": 1,
193             "stakeFlag": true,
194             "_whitelistAdmins": {
195               "bearer": [
196                 {
```

```
            "key": 0,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ]
      },
      "lockReason": [
        {
          "key": "ALL_OTHERS",
          "value": [
            "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
            "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
            "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
            "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF",
            "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
          ]
        }
      ],
      "locked": [
        {
          "key": "ALL_OTHERS",
          "value": [
            {
              "key": "ALL_OTHERS",
              "value": {
                "amount": 5,
                "validity": 5,
                "claimed": false
              }
            }
          ]
        }
      ],
      "_paused": false,
      "_balances": [
        {
          "key": 2,
          "value": 68
        },
        {
          "key": 0,
          "value": 4
        },
        {
          "key": "ALL_OTHERS",
          "value": 5
        }
      ],
      "_pausers": {
        "bearer": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ]
```

```
          },
          "_adminList": [],
          "_allowances": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 5
                }
              ]
            }
          ],
          "_totalSupply": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

Formal Verification Request 56

stakeOf

📅 20, Jan 2020
⏱ 8.13 ms

Line 170-172 in File StakingToken.sol

```
/*@CTK "stakeOf"
  @post __return == stakes[_stakeholder]
*/
```

Line 173-175 in File StakingToken.sol

```
function stakeOf(address _stakeholder) public view returns(uint256) {
  return stakes[_stakeholder];
}
```

✅ The code meets the specification.

Formal Verification Request 57

stakeholderList

📅 20, Jan 2020
⏱ 7.16 ms

Line 229-231 in File StakingToken.sol

```
/*@CTK stakeholderList
  @post __return == stakeholders
*/
```

Line 232-234 in File StakingToken.sol

```
232  function stakeholderList() external view returns (address[] memory) {
233    return stakeholders;
234  }
```

The code meets the specification.

Formal Verification Request 58

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

20, Jan 2020
8.73 ms

Line 43 in File LockableToken.sol

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

Line 51-58 in File LockableToken.sol

```
51  function tokensLocked(address _of, bytes32 _reason)
52      public
53      view
54      returns (uint256 amount)
55  {
56    if (!locked[_of][_reason].claimed)
57        amount = locked[_of][_reason].amount;
58  }
```

The code meets the specification.

Formal Verification Request 59

If method completes, integer overflow would not happen.

20, Jan 2020
0.48 ms

Line 44 in File LockableToken.sol

```
44  //@CTK NO_OVERFLOW
```

Line 51-58 in File LockableToken.sol

```
51  function tokensLocked(address _of, bytes32 _reason)
52      public
53      view
54      returns (uint256 amount)
55  {
56    if (!locked[_of][_reason].claimed)
57        amount = locked[_of][_reason].amount;
58  }
```

The code meets the specification.

Formal Verification Request 60

Method will not encounter an assertion failure.

20, Jan 2020
0.43 ms

Line 45 in File LockableToken.sol

```
45  //@CTK NO_ASF
```

Line 51-58 in File LockableToken.sol

```
51  function tokensLocked(address _of, bytes32 _reason)
52      public
53      view
54      returns (uint256 amount)
55  {
56    if (!locked[_of][_reason].claimed)
57        amount = locked[_of][_reason].amount;
58  }
```

The code meets the specification.

Formal Verification Request 61

tokensLocked

20, Jan 2020
2.18 ms

Line 46-50 in File LockableToken.sol

```
46  /*@CTK "tokensLocked"
47    @tag assume_completion
48    @post locked[_of][_reason].claimed -> amount == 0
49    @post !locked[_of][_reason].claimed -> amount == locked[_of][_reason].amount
50  */
```

Line 51-58 in File LockableToken.sol

```
51  function tokensLocked(address _of, bytes32 _reason)
52      public
53      view
54      returns (uint256 amount)
55  {
56    if (!locked[_of][_reason].claimed)
57        amount = locked[_of][_reason].amount;
58  }
```

The code meets the specification.

Formal Verification Request 62

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

20, Jan 2020
9.73 ms

Line 67 in File LockableToken.sol

```
67  //@CTK NO_BUF_OVERFLOW
```

Line 75-82 in File LockableToken.sol

```
75  function tokensLockedAtTime(address _of, bytes32 _reason, uint256 _time)
76      public
77      view
78      returns (uint256 amount)
79  {
80    if (locked[_of][_reason].validity > _time)
81        amount = locked[_of][_reason].amount;
82  }
```

✅ The code meets the specification.

Formal Verification Request 63

If method completes, integer overflow would not happen.

📅 20, Jan 2020
⏱ 1.31 ms

Line 68 in File LockableToken.sol

```
68  //@CTK NO_OVERFLOW
```

Line 75-82 in File LockableToken.sol

```
75  function tokensLockedAtTime(address _of, bytes32 _reason, uint256 _time)
76      public
77      view
78      returns (uint256 amount)
79  {
80    if (locked[_of][_reason].validity > _time)
81        amount = locked[_of][_reason].amount;
82  }
```

✅ The code meets the specification.

Formal Verification Request 64

Method will not encounter an assertion failure.

📅 20, Jan 2020
⏱ 0.87 ms

Line 69 in File LockableToken.sol

```
69  //@CTK NO_ASF
```

Line 75-82 in File LockableToken.sol

```
75  function tokensLockedAtTime(address _of, bytes32 _reason, uint256 _time)
76      public
77      view
```

```
78      returns (uint256 amount)
79  {
80    if (locked[_of][_reason].validity > _time)
81        amount = locked[_of][_reason].amount;
82  }
```

The code meets the specification.

Formal Verification Request 65

tokensLockedAtTime

20, Jan 2020
1.96 ms

Line 70-74 in File LockableToken.sol

```
70  /*@CTK "tokensLockedAtTime"
71    @tag assume_completion
72    @post locked[_of][_reason].validity <= _time -> amount == 0
73    @post locked[_of][_reason].validity > _time -> amount == locked[_of][_reason].amount
74  */
```

Line 75-82 in File LockableToken.sol

```
75  function tokensLockedAtTime(address _of, bytes32 _reason, uint256 _time)
76      public
77      view
78      returns (uint256 amount)
79  {
80    if (locked[_of][_reason].validity > _time)
81        amount = locked[_of][_reason].amount;
82  }
```

The code meets the specification.

Formal Verification Request 66

tokensValidity

20, Jan 2020
7.49 ms

Line 90-93 in File LockableToken.sol

```
90  /*@CTK "tokensValidity"
91    @tag assume_completion
92    @post validity == locked[_of][_reason].validity
93  */
```

Line 94-100 in File LockableToken.sol

```
94  function tokensValidity(address _of, bytes32 _reason)
95      public
96      view
97      returns (uint256 validity)
98  {
```

```
99    validity = locked[_of][_reason].validity;
100 }
```

The code meets the specification.

Formal Verification Request 67

Method will not encounter an assertion failure.

20, Jan 2020
14.83 ms

Line 106 in File LockableToken.sol

```
106 //@CTK NO_ASF
```

Line 107-126 in File LockableToken.sol

```
107 function lockBalanceOf(address _of)
108     public
109     view
110     returns (uint256 amount)
111 {
112   amount = 0;
113
114   uint256 i = 0;
115
116   /*@CTK getTotallockedBalance
117     @inv _of == _of__pre
118     @inv i <= this.lockReason[_of].length
119     @inv amount >= amount
120     @post i == this.lockReason[_of].length
121     @post !__should_return
122   */
123   for ( i ; i < lockReason[_of].length; i++) {
124     amount = amount.add(tokensLocked(_of, lockReason[_of][i]));
125   }
126 }
```

The code meets the specification.

Formal Verification Request 68

tokensUnlockable

20, Jan 2020
15.4 ms

Line 133-137 in File LockableToken.sol

```
133 /*@CTK tokensUnlockable
134   @tag assume_completion
135   @post (locked[_of][_reason].validity > now || locked[_of][_reason].claimed) -> amount ==
          0
136   @post (locked[_of][_reason].validity <= now && !locked[_of][_reason].claimed) -> amount
        == locked[_of][_reason].amount
137 */
```

Line 138-145 in File LockableToken.sol

```
138 function tokensUnlockable(address _of, bytes32 _reason)
139     public
140     view
141     returns (uint256 amount)
142 {
143     if (locked[_of][_reason].validity <= now && !locked[_of][_reason].claimed) //solium-
            disable-line
144         amount = locked[_of][_reason].amount;
145 }
```

The code meets the specification.

Formal Verification Request 69

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

20, Jan 2020
137.43 ms

Line 151 in File LockableToken.sol

```
151 //@CTK NO_BUF_OVERFLOW
```

Line 154-183 in File LockableToken.sol

```
154 function unlock(address _of)
155     public
156     isAdminOrSelf(_of)
157     returns (uint256 unlockableTokens)
158 {
159   uint256 lockedTokens;
160   uint256 i = 0;
161   /*#CTK "Forloop_unlockAll"
162     @inv i <= lockReason[_of].length
163     @inv forall j: uint. (j >= 0 /\ j < i /\ (locked[_of][lockReason[_of][i]].validity <=
          now && !locked[_of][lockReason[_of][i]].claimed && locked[_of][lockReason[_of][i]].
          amount > 0)) -> locked[_of][lockReason[_of][i]].claimed
164     @post i == lockReason[_of].length
165     @post !__should_return
166   */
167   /*@CTK loop
168     @inv true
169   */
170   for ( i ; i < lockReason[_of].length; i++) {
171     lockedTokens = tokensUnlockable(_of, lockReason[_of][i]);
172     if (lockedTokens > 0) {
173       unlockableTokens = unlockableTokens.add(lockedTokens);
174       locked[_of][lockReason[_of][i]].claimed = true;
175       emit Unlocked(_of, lockReason[_of][i], lockedTokens);
176     }
177   }
178   if (unlockableTokens > 0) {
179     _transfer(address(this), _of, unlockableTokens);
180   }
181 }
```

The code meets the specification.

Formal Verification Request 70

If method completes, integer overflow would not happen.

20, Jan 2020
1.56 ms

Line 152 in File LockableToken.sol

```
152  //@CTK NO_OVERFLOW
```

Line 154-183 in File LockableToken.sol

```
154  function unlock(address _of)
155      public
156      isAdminOrSelf(_of)
157      returns (uint256 unlockableTokens)
158  {
159    uint256 lockedTokens;
160    uint256 i = 0;
161    /*#CTK "Forloop_unlockAll"
162      @inv i <= lockReason[_of].length
163      @inv forall j: uint. (j >= 0 /\ j < i /\ (locked[_of][lockReason[_of][i]].validity <=
           now && !locked[_of][lockReason[_of][i]].claimed && locked[_of][lockReason[_of][i]].
           amount > 0)) -> locked[_of][lockReason[_of][i]].claimed
164      @post i == lockReason[_of].length
165      @post !__should_return
166    */
167    /*@CTK loop
168      @inv true
169    */
170    for ( i ; i < lockReason[_of].length; i++) {
171      lockedTokens = tokensUnlockable(_of, lockReason[_of][i]);
172      if (lockedTokens > 0) {
173        unlockableTokens = unlockableTokens.add(lockedTokens);
174        locked[_of][lockReason[_of][i]].claimed = true;
175        emit Unlocked(_of, lockReason[_of][i], lockedTokens);
176      }
177    }
178    if (unlockableTokens > 0) {
179      _transfer(address(this), _of, unlockableTokens);
180    }
181  }
```

The code meets the specification.

Formal Verification Request 71

Method will not encounter an assertion failure.

20, Jan 2020
1.2 ms

Line 153 in File LockableToken.sol

```
153  //@CTK NO_ASF
```

Line 154-183 in File LockableToken.sol

```
154  function unlock(address _of)
155      public
156      isAdminOrSelf(_of)
157      returns (uint256 unlockableTokens)
158  {
159    uint256 lockedTokens;
160    uint256 i = 0;
161    /*#CTK "Forloop_unlockAll"
162      @inv i <= lockReason[_of].length
163      @inv forall j: uint. (j >= 0 /\ j < i /\ (locked[_of][lockReason[_of][i]].validity <=
           now && !locked[_of][lockReason[_of][i]].claimed && locked[_of][lockReason[_of][i]].
           amount > 0)) -> locked[_of][lockReason[_of][i]].claimed
164      @post i == lockReason[_of].length
165      @post !__should_return
166    */
167    /*@CTK loop
168      @inv true
169    */
170    for ( i ; i < lockReason[_of].length; i++) {
171      lockedTokens = tokensUnlockable(_of, lockReason[_of][i]);
172      if (lockedTokens > 0) {
173        unlockableTokens = unlockableTokens.add(lockedTokens);
174        locked[_of][lockReason[_of][i]].claimed = true;
175        emit Unlocked(_of, lockReason[_of][i], lockedTokens);
176      }
177    }
178    if (unlockableTokens > 0) {
179      _transfer(address(this), _of, unlockableTokens);
180    }
181  }
```

The code meets the specification.

Formal Verification Request 72

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

20, Jan 2020
13.32 ms

Line 189 in File LockableToken.sol

```
189  //@CTK NO_BUF_OVERFLOW
```

Line 192-208 in File LockableToken.sol

```
192  function getUnlockableTokens(address _of)
193      public
194      view
195      returns (uint256 unlockableTokens)
196  {
197    uint256 i = 0;
198    /*@CTK Forloop_getUnlockableTokens
199      @inv _of == _of__pre
200      @inv i <= this.lockReason[_of].length
201      @inv unlockableTokens >= unlockableTokens
202      @post i == this.lockReason[_of].length
203      @post !__should_return
```

```
204     */
205     for ( i ; i < lockReason[_of].length; i++) {
206       unlockableTokens = unlockableTokens.add(tokensUnlockable(_of, lockReason[_of][i]));
207     }
208   }
```

The code meets the specification.

Formal Verification Request 73

If method completes, integer overflow would not happen.

20, Jan 2020
0.47 ms

Line 190 in File LockableToken.sol

```
190   //@CTK NO_OVERFLOW
```

Line 192-208 in File LockableToken.sol

```
192   function getUnlockableTokens(address _of)
193       public
194       view
195       returns (uint256 unlockableTokens)
196   {
197     uint256 i = 0;
198     /*@CTK Forloop_getUnlockableTokens
199       @inv _of == _of__pre
200       @inv i <= this.lockReason[_of].length
201       @inv unlockableTokens >= unlockableTokens
202       @post i == this.lockReason[_of].length
203       @post !__should_return
204     */
205     for ( i ; i < lockReason[_of].length; i++) {
206       unlockableTokens = unlockableTokens.add(tokensUnlockable(_of, lockReason[_of][i]));
207     }
208   }
```

The code meets the specification.

Formal Verification Request 74

Method will not encounter an assertion failure.

20, Jan 2020
0.49 ms

Line 191 in File LockableToken.sol

```
191   //@CTK NO_ASF
```

Line 192-208 in File LockableToken.sol

```
192   function getUnlockableTokens(address _of)
193       public
194       view
```

```
195     returns (uint256 unlockableTokens)
196   {
197     uint256 i = 0;
198     /*@CTK Forloop_getUnlockableTokens
199       @inv _of == _of__pre
200       @inv i <= this.lockReason[_of].length
201       @inv unlockableTokens >= unlockableTokens
202       @post i == this.lockReason[_of].length
203       @post !__should_return
204     */
205     for ( i ; i < lockReason[_of].length; i++) {
206       unlockableTokens = unlockableTokens.add(tokensUnlockable(_of, lockReason[_of][i]));
207     }
208   }
```

The code meets the specification.

Formal Verification Request 75

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

20, Jan 2020
796.65 ms

Line 218 in File LockableToken.sol

```
218   //@CTK NO_BUF_OVERFLOW
```

Line 228-246 in File LockableToken.sol

```
228   function transferWithLock(address _to, bytes32 _reason, uint256 _amount, uint256 _time)
          public onlyWhitelistAdmin returns (bool) {
229     require(_to != address(0), "Zero address not allowed");
230     require(_amount != 0, AMOUNT_ZERO);
231     require(tokensLocked(_to, _reason) == 0, ALREADY_LOCKED);
232     require(balanceOf(msg.sender) > _amount);
233
234     uint256 validUntil = now.add(_time); //solium-disable-line
235
236     // not allowed duplicate reason for address
237     if (locked[_to][_reason].amount == 0)
238       lockReason[_to].push(_reason);
239
240     _transfer(msg.sender, address(this), _amount);
241
242     locked[_to][_reason] = LockToken(_amount, validUntil, false);
243
244     emit Locked(_to, _reason, _amount, validUntil);
245     return true;
246   }
```

The code meets the specification.

Formal Verification Request 76

Method will not encounter an assertion failure.

20, Jan 2020

50.55 ms

Line 219 in File LockableToken.sol

```
219  //@CTK NO_ASF
```

Line 228-246 in File LockableToken.sol

```
228  function transferWithLock(address _to, bytes32 _reason, uint256 _amount, uint256 _time)
         public onlyWhitelistAdmin returns (bool) {
229    require(_to != address(0), "Zero address not allowed");
230    require(_amount != 0, AMOUNT_ZERO);
231    require(tokensLocked(_to, _reason) == 0, ALREADY_LOCKED);
232    require(balanceOf(msg.sender) > _amount);
233
234    uint256 validUntil = now.add(_time); //solium-disable-line
235
236    // not allowed duplicate reason for address
237    if (locked[_to][_reason].amount == 0)
238      lockReason[_to].push(_reason);
239
240    _transfer(msg.sender, address(this), _amount);
241
242    locked[_to][_reason] = LockToken(_amount, validUntil, false);
243
244    emit Locked(_to, _reason, _amount, validUntil);
245    return true;
246  }
```

The code meets the specification.

Formal Verification Request 77

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

20, Jan 2020
259.24 ms

Line 254 in File LockableToken.sol

```
254  //@CTK NO_BUF_OVERFLOW
```

Line 263-271 in File LockableToken.sol

```
263  function extendLock(address _to, bytes32 _reason, uint256 _time) public onlyWhitelistAdmin
         returns (bool) {
264    require(_to != address(0), "Zero address not allowed");
265    require(tokensLocked(_to, _reason) > 0, NOT_LOCKED);
266
267    locked[_to][_reason].validity = locked[_to][_reason].validity.add(_time);
268
269    emit Locked(_to, _reason, locked[_to][_reason].amount, locked[_to][_reason].validity);
270    return true;
271  }
```

The code meets the specification.

Formal Verification Request 78

If method completes, integer overflow would not happen.

20, Jan 2020
41.59 ms

Line 255 in File LockableToken.sol

```
255  //@CTK NO_OVERFLOW
```

Line 263-271 in File LockableToken.sol

```
263  function extendLock(address _to, bytes32 _reason, uint256 _time) public onlyWhitelistAdmin
         returns (bool) {
264    require(_to != address(0), "Zero address not allowed");
265    require(tokensLocked(_to, _reason) > 0, NOT_LOCKED);
266
267    locked[_to][_reason].validity = locked[_to][_reason].validity.add(_time);
268
269    emit Locked(_to, _reason, locked[_to][_reason].amount, locked[_to][_reason].validity);
270    return true;
271  }
```

The code meets the specification.

Formal Verification Request 79

Method will not encounter an assertion failure.

20, Jan 2020
21.14 ms

Line 256 in File LockableToken.sol

```
256  //@CTK NO_ASF
```

Line 263-271 in File LockableToken.sol

```
263  function extendLock(address _to, bytes32 _reason, uint256 _time) public onlyWhitelistAdmin
         returns (bool) {
264    require(_to != address(0), "Zero address not allowed");
265    require(tokensLocked(_to, _reason) > 0, NOT_LOCKED);
266
267    locked[_to][_reason].validity = locked[_to][_reason].validity.add(_time);
268
269    emit Locked(_to, _reason, locked[_to][_reason].amount, locked[_to][_reason].validity);
270    return true;
271  }
```

The code meets the specification.

Formal Verification Request 80

extendLock

20, Jan 2020
127.42 ms

Line 257-262 in File LockableToken.sol

```
257 /*@CTK extendLock
258   @tag assume_completion
259   @pre (!locked[_to][_reason].claimed && locked[_to][_reason].amount > 0)
260   @post !_whitelistAdmins.bearer[msg.sender] -> __reverted
261   @post __post.locked[_to][_reason].validity == locked[_to][_reason].validity + _time
262 */
```

Line 263-271 in File LockableToken.sol

```
263 function extendLock(address _to, bytes32 _reason, uint256 _time) public onlyWhitelistAdmin
        returns (bool) {
264   require(_to != address(0), "Zero address not allowed");
265   require(tokensLocked(_to, _reason) > 0, NOT_LOCKED);
266
267   locked[_to][_reason].validity = locked[_to][_reason].validity.add(_time);
268
269   emit Locked(_to, _reason, locked[_to][_reason].amount, locked[_to][_reason].validity);
270   return true;
271 }
```

✅ The code meets the specification.

Formal Verification Request 81

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

📅 20, Jan 2020
⏱ 405.51 ms

Line 279 in File LockableToken.sol

```
279 //@CTK NO_BUF_OVERFLOW
```

Line 290-299 in File LockableToken.sol

```
290 function increaseLockAmount(address _to, bytes32 _reason, uint256 _amount) public
        onlyWhitelistAdmin returns (bool) {
291   require(_to != address(0), "Zero address not allowed");
292   require(tokensLocked(_to, _reason) > 0, NOT_LOCKED);
293   _transfer(msg.sender, address(this), _amount);
294
295   locked[_to][_reason].amount = locked[_to][_reason].amount.add(_amount);
296
297   emit Locked(_to, _reason, locked[_to][_reason].amount, locked[_to][_reason].validity);
298   return true;
299 }
```

✅ The code meets the specification.

Formal Verification Request 82

If method completes, integer overflow would not happen.

📅 20, Jan 2020
⏱ 90.9 ms

Line 280 in File LockableToken.sol

```
280  //@CTK NO_OVERFLOW
```

Line 290-299 in File LockableToken.sol

```
290  function increaseLockAmount(address _to, bytes32 _reason, uint256 _amount) public
         onlyWhitelistAdmin returns (bool) {
291    require(_to != address(0), "Zero address not allowed");
292    require(tokensLocked(_to, _reason) > 0, NOT_LOCKED);
293    _transfer(msg.sender, address(this), _amount);
294
295    locked[_to][_reason].amount = locked[_to][_reason].amount.add(_amount);
296
297    emit Locked(_to, _reason, locked[_to][_reason].amount, locked[_to][_reason].validity);
298    return true;
299  }
```

The code meets the specification.

Formal Verification Request 83

Method will not encounter an assertion failure.

20, Jan 2020
46.0 ms

Line 281 in File LockableToken.sol

```
281  //@CTK NO_ASF
```

Line 290-299 in File LockableToken.sol

```
290  function increaseLockAmount(address _to, bytes32 _reason, uint256 _amount) public
         onlyWhitelistAdmin returns (bool) {
291    require(_to != address(0), "Zero address not allowed");
292    require(tokensLocked(_to, _reason) > 0, NOT_LOCKED);
293    _transfer(msg.sender, address(this), _amount);
294
295    locked[_to][_reason].amount = locked[_to][_reason].amount.add(_amount);
296
297    emit Locked(_to, _reason, locked[_to][_reason].amount, locked[_to][_reason].validity);
298    return true;
299  }
```

The code meets the specification.

Formal Verification Request 84

increaseLockAmount

20, Jan 2020
188.43 ms

Line 282-289 in File LockableToken.sol

```
/*@CTK increaseLockAmount
  @tag assume_completion
  @pre _to != msg.sender
  @post !_whitelistAdmins.bearer[msg.sender] -> __reverted
  @post (locked[_to][_reason].claimed || locked[_to][_reason].amount == 0 ) -> __reverted
  @post (!locked[_to][_reason].claimed && locked[_to][_reason].amount > 0) ->
    __post.locked[_to][_reason].amount == locked[_to][_reason].amount + _amount
*/
```

Line 290-299 in File LockableToken.sol

```
function increaseLockAmount(address _to, bytes32 _reason, uint256 _amount) public
    onlyWhitelistAdmin returns (bool) {
  require(_to != address(0), "Zero address not allowed");
  require(tokensLocked(_to, _reason) > 0, NOT_LOCKED);
  _transfer(msg.sender, address(this), _amount);

  locked[_to][_reason].amount = locked[_to][_reason].amount.add(_amount);

  emit Locked(_to, _reason, locked[_to][_reason].amount, locked[_to][_reason].validity);
  return true;
}
```

✅ The code meets the specification.

Formal Verification Request 85

getSizeOfLockReason

📅 20, Jan 2020
⏱ 12.38 ms

Line 306-309 in File LockableToken.sol

```
/*@CTK getSizeOfLockReason
  @tag assume_completion
  @post __return == lockReason[_of].length
*/
```

Line 310-312 in File LockableToken.sol

```
function getSizeOfLockReason(address _of) public view returns (uint256) {
  return lockReason[_of].length;
}
```

✅ The code meets the specification.

Formal Verification Request 86

getTotallockedBalance___Generated

📅 20, Jan 2020
⏱ 357.24 ms

(Loop) Line 116-122 in File LockableToken.sol

```
116   /*@CTK getTotallockedBalance
117     @inv _of == _of__pre
118     @inv i <= this.lockReason[_of].length
119     @inv amount >= amount
120     @post i == this.lockReason[_of].length
121     @post !__should_return
122   */
```

(Loop) Line 116-125 in File LockableToken.sol

```
116   /*@CTK getTotallockedBalance
117     @inv _of == _of__pre
118     @inv i <= this.lockReason[_of].length
119     @inv amount >= amount
120     @post i == this.lockReason[_of].length
121     @post !__should_return
122   */
123   for ( i ; i < lockReason[_of].length; i++) {
124     amount = amount.add(tokensLocked(_of, lockReason[_of][i]));
125   }
```

The code meets the specification.

Formal Verification Request 87

loop___Generated

20, Jan 2020
205.99 ms

(Loop) Line 167-169 in File LockableToken.sol

```
167   /*@CTK loop
168     @inv true
169   */
```

(Loop) Line 167-177 in File LockableToken.sol

```
167   /*@CTK loop
168     @inv true
169   */
170   for ( i ; i < lockReason[_of].length; i++) {
171     lockedTokens = tokensUnlockable(_of, lockReason[_of][i]);
172     if (lockedTokens > 0) {
173       unlockableTokens = unlockableTokens.add(lockedTokens);
174       locked[_of][lockReason[_of][i]].claimed = true;
175       emit Unlocked(_of, lockReason[_of][i], lockedTokens);
176     }
177   }
```

The code meets the specification.

Formal Verification Request 88

Forloop_getUnlockableTokens___Generated

20, Jan 2020
114.64 ms

(Loop) Line 198-204 in File LockableToken.sol

```
198   /*@CTK Forloop_getUnlockableTokens
199     @inv _of == _of__pre
200     @inv i <= this.lockReason[_of].length
201     @inv unlockableTokens >= unlockableTokens
202     @post i == this.lockReason[_of].length
203     @post !__should_return
204   */
```

(Loop) Line 198-207 in File LockableToken.sol

```
198   /*@CTK Forloop_getUnlockableTokens
199     @inv _of == _of__pre
200     @inv i <= this.lockReason[_of].length
201     @inv unlockableTokens >= unlockableTokens
202     @post i == this.lockReason[_of].length
203     @post !__should_return
204   */
205   for ( i ; i < lockReason[_of].length; i++) {
206     unlockableTokens = unlockableTokens.add(tokensUnlockable(_of, lockReason[_of][i]));
207   }
```

✅ The code meets the specification.

Formal Verification Request 89

_addWhitelistAdmin

📅 20, Jan 2020
⏱ 72.81 ms

Line 17-21 in File AdminRole.sol

```
17  /*@CTK _addWhitelistAdmin
18    @tag assume_completion
19    @post __post._adminList.length == _adminList.length + 1
20    @post __post._adminList[_adminList.length] == account
21  */
```

Line 22-27 in File AdminRole.sol

```
22  function _addWhitelistAdmin(address account) internal {
23    super._addWhitelistAdmin(account);
24    _adminList.push(account);
25  }
```

✅ The code meets the specification.

Source Code with CertiK Labels

File RewardSupplier.sol

```
1  pragma solidity ^0.5.6;
2
3  import "./token/HiblocksIERC20.sol";
4  import "@openzeppelin/contracts/math/SafeMath.sol";
5
6  contract RewardSupplier {
7      using SafeMath for uint256;
8      /**
9      * @dev ERC20 Token under lockable functionality
10     */
11     HiblocksIERC20 private _token;
12
13     /**
14     * @dev constant of reward
15     */
16     struct RewardConst {
17         uint256 totalClaimed; // all claimed reward
18         uint256 totalReward; // total reward for hiblocks platform
19         uint256 releaseTime; // timestamp when token release is enabled
20         address beneficiary; // beneficiary of tokens after they are unlocked
21         uint256 factor; // factor value for calculate reward
22     }
23
24     /**
25     * @dev claimed token structure
26     */
27     struct ClaimedToken {
28         uint256 amount;
29         uint256 week;
30         uint256 released;
31     }
32
33     RewardConst private _const;
34     uint256 private _secondsOfWeek;
35     uint256 private _secondsOfTenYears;
36     uint256 private _decimalLength;
37
38     /**
39     * @dev record the reward tokens sent
40     */
41     ClaimedToken[] private _claimedTokens;
42
43     event RewardLocked(address beneficiary, uint256 releaseTime);
44     event TokenWithdraw(address tokenAddr, address indexed sendTo, uint256 amount);
45
46     /**
47      * @dev Check for administrator privileges.
48      */
49     modifier onlyWhitelistAdmin() {
50         require(_token.isWhitelistAdmin(msg.sender), "WhitelistAdminRole: caller does not
            have the WhitelistAdmin role");
51         _;
52
53     }
```

```
54
55      /**
56      * @dev constructor to lockable initial tokens
57      * @param tokenAddr address of hiblocks tokens
58      * @param beneficiary address of beneficiary
59      * @param releaseTime start time of release
60      */
61      constructor(address tokenAddr, address beneficiary, uint256 totalReward, uint256
            releaseTime) public {
62          _token = HiblocksIERC20(tokenAddr);
63          _secondsOfWeek = 60 * 60 * 24 * 7;
64          _secondsOfTenYears = 60 * 60 * 24 * 365 * 10;
65          _decimalLength = 10 ** 18;
66
67          _const = RewardConst(uint256(0), totalReward, releaseTime, beneficiary, _factor());
68
69          emit RewardLocked(beneficiary, releaseTime);
70      }
71
72      /**
73      * @return factor value for calculate increased reward
74      */
75      //@CTK NO_BUF_OVERFLOW
76      //@CTK NO_OVERFLOW
77      //@CTK NO_ASF
78      function _factor() internal pure returns (uint256 factor) {
79          // uint256 x1 = 1; // first week
80          // uint256 y1 = 3000000; // reward tokens for first week
81          // uint256 x2 = 52 * 2; // last week of 2 years
82          // uint256 y2 = 16800000; // reward tokens for last week
83
84          // factor = (y2.sub(y1)).div((x2.sub(x1)).mul((x2.sub(x1))));
85          factor = 1300;
86      }
87
88      /**
89      * @return the token being held.
90      */
91      /*@CTK token
92        @post __return == _token
93      */
94      function token() external view returns (HiblocksIERC20) {
95          return _token;
96      }
97
98      /**
99      * @return the releaseTime of the tokens.
100     */
101     /*@CTK releaseTime
102       @post __return == _const.releaseTime
103     */
104     function releaseTime() external view returns (uint256) {
105         return _const.releaseTime;
106     }
107
108     /**
109     * @return the beneficiary of the tokens.
110     */
```

```
111     /*@CTK beneficiary
112       @post __return == _const.beneficiary
113     */
114     function beneficiary() external view returns (address) {
115         return _const.beneficiary;
116     }
117 
118     /**
119     * @return total reward for hiblocks platform
120     */
121     /*@CTK beneficiary
122       @post __return == _const.totalReward * (10**18)
123     */
124     function totalReward() external view returns (uint256) {
125         return (_const.totalReward.mul(10**18));
126     }
127 
128     /**
129     * @return remain reward for hiblocks platform
130     */
131     function remainReward() public view returns (uint256) {
132         return _token.balanceOf(address(this));
133     }
134 
135     /**
136     * @return the time when all reward tokens are released.
137     */
138     //@CTK NO_BUF_OVERFLOW
139     //@CTK NO_OVERFLOW
140     //@CTK NO_ASF
141     /*@CTK "endTime"
142       @tag assume_completion
143       @post __return == _const.releaseTime + _secondsOfTenYears
144     */
145     function endTime() public view returns (uint256) {
146         return _const.releaseTime.add(_secondsOfTenYears); // end time of reward
                distribution;
147     }
148 
149     /**
150     * @dev Returns unlockable tokens for a specified address for a specified reason
151     * @param week query the unlockable token count of week
152     */
153     function tokensUnlockableUntil(uint256 week)
154         public view
155         returns (uint256)
156     {
157         uint256 i = _claimedTokens.length == 0 ? 0 : _claimedTokens[_claimedTokens.length-1].
                week + 1;
158         uint256 total = 0;
159         for ( i ; i <= week; i++) {
160             total = total.add(tokensUnlockableAt(i));
161         }
162         return total;
163     }
164 
165     /**
166     * @dev Unlockable tokens at given week
```

```
167     * @param week query the unlockable token count of week
168     */
169     /*#CTK "tokensUnlockableAt"
170       @tag assume_completion
171       @post week < 104 -> __return == (_const.factor * (week ** 2) + 3000000) * (10**18)
172       @post week < 520 -> __return == (_token._balances / 10 **18))/ (521 - week) * (10**18)
173     */
174     function tokensUnlockableAt(uint256 week)
175         public view
176         returns (uint256)
177     {
178         if(week < 104) {
179             return (_const.factor.mul((week) ** 2).add(3000000)).mul(_decimalLength);
180         } else if(week < 520) {
181             uint256 reward = remainReward().div(_decimalLength);
182             return reward.div(521 - week).mul(_decimalLength);
183         }
184         return remainReward();
185     }
186
187     /**
188     * @dev current week of the 10 years
189     */
190     //@CTK NO_BUF_OVERFLOW
191     //@CTK NO_OVERFLOW
192     //@CTK NO_ASF
193     /*@CTK "thisWeek"
194       @tag assume_completion
195       @post _const.releaseTime >= now -> __reverted
196       @post __return == (now - _const.releaseTime) / _secondsOfWeek
197     */
198     function thisWeek() public view returns (uint256){
199         require(_const.releaseTime < now, "not a release period."); //solium-disable-line
200         return now.sub(_const.releaseTime).div(_secondsOfWeek); //solium-disable-line
201     }
202
203     /*#CTK unlockReward
204       @tag assume_completion
205       @post _const.releaseTime >= now -> __reverted
206       @post _token._balance == 0 -> __reverted
207     */
208     function unlockReward() external returns (bool) {
209         require(_const.releaseTime < now, "not a release period."); //solium-disable-line
210         require(remainReward() > 0, "Insufficient balance");
211
212         uint256 week = thisWeek();
213         require(_claimedTokens.length == 0 || _claimedTokens[_claimedTokens.length-1].week <
                week, "token already claimed.");
214
215         uint256 reward = tokensUnlockableUntil(week);
216         if(reward > 0){
217             _token.transfer(_const.beneficiary, reward);
218             _const.totalClaimed.add(reward);
219             _claimedTokens.push(ClaimedToken(reward, week, now)); //solium-disable-line
220
221             emit TokenWithdraw(address(_token), _const.beneficiary, reward);
222         }
223         return true;
```

```
224
225     }
226
227
228
229     /**
230     * @dev Change new beneficiary
231     * @param newBeneficiary address of new beneficiary
232     */
233     //@CTK NO_BUF_OVERFLOW
234     //@CTK NO_OVERFLOW
235     //@CTK NO_ASF
236     /*@CTK "setBeneficiary"
237       @post _const.beneficiary == newBeneficiary -> __return == true
238     */
239     function setBeneficiary(address newBeneficiary) external onlyWhitelistAdmin returns (
            bool) {
240         _const.beneficiary = newBeneficiary;
241         emit RewardLocked(newBeneficiary, _const.releaseTime);
242         return true;
243     }
244
245     /**
246     * @dev extend release time.
247     * @param extendTime seconds to add
248     */
249     //@CTK NO_BUF_OVERFLOW
250     //@CTK NO_OVERFLOW
251     //@CTK NO_ASF
252     /*#CTK extendReleaseTime
253       @post _const.releaseTime >= now -> __reverted
254       @post _const.releaseTime == _const.releaseTime + extendTime
255       @post __return == true
256     */
257     function extendReleaseTime(uint256 extendTime) external onlyWhitelistAdmin returns (bool
            ) {
258         require(_const.releaseTime > now, "release time is in past"); //solium-disable-line
259
260         _const.releaseTime = _const.releaseTime.add(extendTime);
261         emit RewardLocked(_const.beneficiary, _const.releaseTime);
262         return true;
263     }
264
265     /**
266     * @dev withdrawal the remain balance after the compensation period ends.
267     */
268     /*#CTK withdrawalRemain
269       @post _const.releaseTime >= now -> __reverted
270       @post __return == true
271     */
272     function withdrawalRemain() external onlyWhitelistAdmin returns (bool) {
273         require(endTime() < now, "Reward period is left."); //solium-disable-line
274         require(remainReward() > 0, "Insufficient balance");
275
276         _token.transfer(_const.beneficiary, remainReward());
277         emit TokenWithdraw(address(_token), _const.beneficiary, remainReward());
278         return true;
279     }
```

```
280 }
```

File HiblocksToken.sol

```
1   pragma solidity ^0.5.6;
2
3   import "@openzeppelin/contracts/token/ERC20/ERC20Detailed.sol";
4   import "./token/StakingToken.sol";
5
6   contract HiblocksToken is ERC20Detailed, StakingToken {
7
8     mapping(address => mapping(uint => bytes)) public memos;
9
10    /**
11    * @dev constructor to mint initial tokens
12    * @param name string
13    * @param symbol string
14    * @param decimals uint8
15    * @param initialSupply uint256
16    */
17    /*@CTK "constructor"
18      @post initialSupply <= 0 -> __reverted
19    */
20    constructor(string memory name, string memory symbol, uint8 decimals, uint256
          initialSupply)
21      public
22      ERC20Detailed(name, symbol, decimals)
23    {
24      // Mint the initial supply
25      require(initialSupply > 0, "initialSupply must be greater than zero.");
26      _mint(_msgSender(), initialSupply * (10 ** uint256(decimals)));
27    }
28
29    /**
30    * @dev Transfer token for a specified address with memo
31    * @param to The address to transfer to.
32    * @param value The amount to be transferred.
33    * @param memo The memo to be saved.
34    */
35    /*@CTK "transferWithMemo"
36      @tag assume_completion
37      @post __post.memos[to][block.number] == memo
38      @post __return == true
39    */
40    function transferWithMemo(address to, uint256 value, bytes memory memo) public returns (
          bool) {
41      require(transfer(to, value), "Token transfer failed");
42      memos[to][block.number] = memo;
43      return true;
44    }
45
46    /**
47    * @dev Gets the memo of the specified address and block number.
48    * @param addr The address to query the memo of.
49    * @param blockNumber The block number to query the memo of.
50    * @return An bytes representing the memo writed by the passed address.
51    */
52    /*@CTK "memoOf"
53      @post __return == memos[addr][blockNumber]
```

```
54    */
55    function memoOf(address addr, uint blockNumber) external view returns (bytes memory) {
56      return(memos[addr][blockNumber]);
57    }
58
59    /**
60    * @dev Destroys `amount` tokens from the caller.
61    *
62    * See {ERC20-_burn}.
63    */
64    function burn(uint256 amount) public onlyWhitelistAdmin {
65      super._burn(_msgSender(), amount);
66    }
67
68    /**
69    * @dev Destroys `amount` tokens from the address.
70    *
71    * See {ERC20-_burnFrom}.
72    */
73    function burnFrom(address addr, uint256 amount) public onlyWhitelistAdmin {
74      super._burnFrom(addr, amount);
75    }
76
77    /**
78    * @dev Returns total tokens held by an address (transferable + locked)
79    * @param _addr The address to query the total balance of
80    */
81    function totalBalanceOf(address _addr) external view returns (uint256 amount, uint256
          token, uint256 locked, uint256 staked) {
82      token = balanceOf(_addr);
83      locked = lockBalanceOf(_addr);
84      staked = stakeOf(_addr);
85      amount = token.add(locked).add(staked);
86    }
87  }
```

File RewardPool.sol

```
1   pragma solidity ^0.5.6;
2
3   import "./token/HiblocksIERC20.sol";
4   import "@openzeppelin/contracts/math/SafeMath.sol";
5
6   contract RewardPool {
7      using SafeMath for uint256;
8      /**
9      * @dev ERC20 Token under lockable functionality
10     */
11     HiblocksIERC20 private _token;
12
13     // rewardToken[address][week] = rewardToken
14     mapping(address => mapping(uint8 => uint256)) public rewardToken;
15
16     /**
17      * @dev Check for administrator privileges.
18      */
19     modifier onlyWhitelistAdmin() {
20        require(_token.isWhitelistAdmin(msg.sender), "WhitelistAdminRole: caller does not
              have the WhitelistAdmin role");
```

```
21          _;
22      }
23
24      /**
25      * @dev constructor of network reward pool
26      * @param tokenAddr address of hiblocks tokens
27      */
28      //@CTK NO_ASF
29      constructor(address tokenAddr) public {
30          _token = HiblocksIERC20(tokenAddr);
31      }
32
33      /**
34      * @return the token being held.
35      */
36      /*@CTK "token"
37        @post __return == _token
38      */
39      function token() external view returns (HiblocksIERC20) {
40          return _token;
41      }
42
43      /**
44       * @dev Returns the amount of tokens owned by `account`.
45       */
46      function rewardBalance() external view returns (uint256) {
47          return _token.balanceOf(address(this));
48      }
49
50      /**
51       * @dev rewarded token for a specified address and reward week of 10 years
52       * @param recipient The address of reward to.
53       * @param week reward week of 10 years.
54       */
55      /*@CTK "rewardOf"
56        @post __return == rewardToken[recipient][week]
57      */
58      function rewardOf(address recipient, uint8 week) external view returns (uint256) {
59          return rewardToken[recipient][week];
60      }
61
62      /**
63       * @dev Moves `amount` tokens from this contract's account to `recipient` with memo.
64       * @param recipient The address to transfer to.
65       * @param amount The amount to be transferred.
66       * @param week reward week of 10 years.
67       * @param memo The memo to be saved.
68       */
69      function payReward(address recipient, uint256 amount, uint8 week, bytes calldata memo)
            external onlyWhitelistAdmin returns (bool) {
70  //        require(week >= 0 && week < 521, "week order out of range(0~520).");
71          require(rewardBalance() >= amount, "transfer amount exceeds balance");
72
73          if(memo.length>0){
74              require(_token.transferWithMemo(recipient, amount, memo), "Failed
                    transferWithMemo.");
75          }else{
76              require(_token.transfer(recipient, amount), "Failed tranfer.");
```

```
77          }
78          rewardToken[recipient][week] = rewardToken[recipient][week].add(amount);
79          return true;
80      }
81  }
```

File token/PausableToken.sol

```
1   pragma solidity ^0.5.6;
2
3   // hack
4   import "../@openzeppelin/contracts/token/ERC20/ERC20.sol";
5   import "../roles/AdminRole.sol";
6
7   // /**
8   // * @title Pausable token
9   // * @dev ERC20 with pausable transfers and allowances.
10  // *
11  // * Useful if you want to stop trades until the end of a crowdsale, or have
12  // * an emergency switch for freezing all token transfers in the event of a large
13  // * bug.
14  // */
15   contract PausableToken is ERC20, AdminRole {
16
17  //   /**
18  //   * @dev Emitted when the pause is triggered by a whitelistAdmin (`account`).
19  //   */
20    event Paused(address account);
21
22    /**
23    * @dev Emitted when the pause is lifted by a whitelistAdmin (`account`).
24    */
25    event Unpaused(address account);
26
27    bool private _paused;
28
29    // hack
30    Roles.Role private _whitelistAdmins;
31    Roles.Role private _pausers;
32    // end hack
33
34  //   /**
35  //   * @dev Initializes the contract in unpaused state. Assigns the whitelistAdmin role
36  //   * to the deployer.
37  //   */
38    /*@CTK "constructor"
39      @post _paused == false
40    */
41    constructor () internal {
42      _paused = false;
43    }
44
45    /**
46    * @dev Returns true if the contract is paused, and false otherwise.
47    */
48    /*@CTK "paused"
49      @post __return == _paused
50    */
51    function paused() public view returns (bool) {
```

```
52    return _paused;
53  }
54
55  /**
56  * @dev Modifier to make a function callable only when the contract is not paused.
57  */
58  modifier whenNotPaused() {
59    require(!_paused, "Pausable: paused");
60    _;
61  }
62
63  /**
64  * @dev Modifier to make a function callable only when the contract is paused.
65  */
66  modifier whenPaused() {
67    require(_paused, "Pausable: not paused");
68    _;
69  }
70
71  /**
72  * @dev Called by a whitelistAdmin to pause, triggers stopped state.
73  */
74  /*@CTK pause
75    @post _paused -> __reverted
76    @post msg.sender == address(0) -> __reverted
77    @post !_whitelistAdmins.bearer[msg.sender] -> __reverted
78    @post !_paused && msg.sender != address(0) && _whitelistAdmins.bearer[msg.sender] &&
          _pausers.bearer[msg.sender]
79        -> __post._paused
80  */
81  function pause() public onlyWhitelistAdmin whenNotPaused {
82    _paused = true;
83    emit Paused(_msgSender());
84  }
85
86  /**
87  * @dev Called by a whitelistAdmin to unpause, returns to normal state.
88  */
89  /*@CTK unpause
90    @post !_paused -> __reverted
91    @post msg.sender == address(0) -> __reverted
92    @post !_whitelistAdmins.bearer[msg.sender] -> __reverted
93    @post _paused && msg.sender != address(0) && _whitelistAdmins.bearer[msg.sender] &&
          _pausers.bearer[msg.sender]
94        -> !__post._paused
95  */
96  function unpause() public onlyWhitelistAdmin whenPaused {
97    _paused = false;
98    emit Unpaused(_msgSender());
99  }
100
101 /**
102 * @dev set modifier is paused to transfer function
103
104 * @param to address
105 * @param value uint256
106 */
107 //@CTK NO_OVERFLOW
```

```
108   //@CTK NO_BUF_OVERFLOW
109   //@CTK NO_ASF
110   /*@CTK "transfer"
111     @tag assume_completion
112     @post (to == address(0) || _paused == true) -> __reverted
113     @post value <= _balances[msg.sender]
114     @post (to != address(0) && to != msg.sender) -> __post._balances[msg.sender] ==
          _balances[msg.sender] - value
115     @post (to != address(0) && to != msg.sender) -> __post._balances[to] == _balances[to] +
          value
116     @post (to != address(0) && to == msg.sender) -> __post._balances[msg.sender] ==
          _balances[msg.sender]
117   */
118   function transfer(address to, uint256 value) public whenNotPaused returns (bool) {
119     return super.transfer(to, value);
120   }
121
122   /**
123   * @dev set modifier is paused to transferFrom function
124
125   * @param from address
126   * @param to address
127   * @param value uint256
128   */
129   //@CTK NO_OVERFLOW
130   //@CTK NO_BUF_OVERFLOW
131   //@CTK NO_ASF
132   /*@CTK "transferFrom"
133     @tag assume_completion
134     @post (_paused || msg.sender == address(0) || from == address(0) || to == address(0)) ->
          __reverted
135     @post from == to -> __post._balances[from] == _balances[from]
136     @post from != to -> __post._balances[from] == _balances[from] - value
137     @post from != to -> __post._balances[to] == _balances[to] + value
138     @post __post._allowances[from][msg.sender] == _allowances[from][msg.sender] - value
139     @post __return == true
140     */
141   function transferFrom(address from, address to, uint256 value) public whenNotPaused
        returns (bool) {
142     return super.transferFrom(from, to, value);
143   }
144
145   /**
146   * @dev set modifier is paused to approve function
147
148   * @param spender address
149   * @param value uint256
150   */
151   //@CTK NO_OVERFLOW
152   //@CTK NO_BUF_OVERFLOW
153   //@CTK NO_ASF
154   /*@CTK "approve"
155     @tag assume_completion
156     @post (_paused || msg.sender == address(0) || spender == address(0)) -> __reverted
157     @post (!_paused || msg.sender != address(0) || spender != address(0)) ->
158       __post._allowances[msg.sender][spender] == value
159     @post __return == true
160     */
```

```
161   function approve(address spender, uint256 value) public whenNotPaused returns (bool) {
162     return super.approve(spender, value);
163   }
164
165   /**
166   * @dev set modifier is paused to increaseAllowance function
167
168   * @param spender address
169   * @param addedValue uint256
170   */
171
172   //@CTK NO_OVERFLOW
173   //@CTK NO_BUF_OVERFLOW
174   //@CTK NO_ASF
175   /*@CTK "increaseAllowance"
176     @tag assume_completion
177     @post (_paused == true || spender == address(0)) -> __reverted
178     @post (_paused == false && spender != address(0)) -> __post._allowances[msg.sender][
          spender] == _allowances[msg.sender][spender] + addedValue
179     @post __return == true
180   */
181   function increaseAllowance(address spender, uint256 addedValue) public whenNotPaused
        returns (bool) {
182     return super.increaseAllowance(spender, addedValue);
183   }
184
185   /**
186   * @dev set modifier is paused to decreaseAllowance function
187
188   * @param spender address
189   * @param subtractedValue uint256
190   */
191   //@CTK NO_OVERFLOW
192   //@CTK NO_BUF_OVERFLOW
193   //@CTK NO_ASF
194   /*@CTK "decreaseAllowance"
195     @tag assume_completion
196     @post (_paused == true || spender == address(0)) -> __reverted
197     @post (_paused == false && spender != address(0)) -> __post._allowances[msg.sender][
          spender] == _allowances[msg.sender][spender] - subtractedValue
198     */
199   function decreaseAllowance(address spender, uint256 subtractedValue) public whenNotPaused
        returns (bool) {
200     return super.decreaseAllowance(spender, subtractedValue);
201   }
202 }
```

File token/StakingToken.sol

```
1  pragma solidity ^0.5.6;
2
3  import "./LockableToken.sol";
4  /**
5   * @title Staking Token (STK)
6   * @author Alberto Cuesta Canada
7   * @dev Implements a basic ERC20 staking token with incentive distribution.
8   */
9  contract StakingToken is LockableToken {
10   /**
```

```
11  * @dev Emitted when user staked token.
12  */
13  event Staked(address account, uint256 amount);
14
15  /**
16  * @dev Emitted when user unstaked token.
17  */
18  event Unstaked(address account, uint256 amount);
19
20   /**
21  * @dev Emitted when user unstaked token.
22  */
23  event validStakePeriod(uint256 _startTime, uint256 _duration, bool stakeFlag);
24
25  /**
26  * @dev We usually require to know who are all the stakeholders.
27  */
28  address[] internal stakeholders;
29
30  /**
31  * @dev The stakes for each stakeholder.
32  */
33  mapping(address => uint256) internal stakes;
34
35  uint256 private secondsInterval;
36  uint256 private validStakeDuration;
37  uint256 private validStakeStartTime;
38  bool private stakeFlag;
39
40  /**
41  * @dev Initializes the contract in stake state.
42  */
43  /*@CTK "constructor"
44    @post __post.secondsInterval == 604800
45    @post __post.validStakeDuration == 32400
46    @post __post.validStakeStartTime == now - 32400
47    @post __post.stakeFlag == false
48  */
49  constructor () internal {
50    secondsInterval = 60*60*24*7; // default : interval 7 days
51    validStakeDuration = 32400; // default : 60*60*9 ( 9hours )
52    validStakeStartTime = now - validStakeDuration; //solium-disable-line
53    stakeFlag = false;
54  }
55
56  // ---------- SET TIME ----------
57  modifier isvalidStakePeriod() {
58    require(stakeFlag, "stake is disabled");
59    require(isAbleToStake(), "It's not valid time to stake"); //solium-disable-line
60    _;
61  }
62
63  // hack
64  string internal constant ALREADY_LOCKED = "Tokens already locked";
65  string internal constant NOT_LOCKED = "No tokens locked";
66  string internal constant AMOUNT_ZERO = "Amount can not be 0";
67  Roles.Role private _whitelistAdmins;
68  // end hack
```

```
69  /**
70  * @dev Set period of able to stake. (It is automatically set to the same day each week.)
71  * @param _startTime uint256 available start time to set stake/unstake
72  * @param _duration uint256 duration second to set stake/unstake
73  */
74  // /*@CTK "setvalidStakePeriod"
75  //   @post !_whitelistAdmins.bearer[msg.sender] -> __reverted
76  //   @post secondsInterval == _startTime
77  //   @post validStakeDuration == _duration
78  //   @post stakeFlag == true
79  // */
80  function setvalidStakePeriod(uint256 _startTime, uint256 _duration) public
        onlyWhitelistAdmin returns (uint256, uint256){
81    validStakeStartTime = _startTime;
82    validStakeDuration = _duration;
83    stakeFlag = true;
84    return stakePeriod();
85  }
86
87  /**
88  * @dev change the stakeFlag state.
89  * @param _stakeFlag bool stakeFlag state
90  */
91
92  function setStakeFlag(bool _stakeFlag) public onlyWhitelistAdmin returns (bool) {
93    stakeFlag = _stakeFlag;
94    return stakeFlag;
95  }
96
97  /**
98  * @dev A method for check is able to stake
99  */
100 /*@CTK "isAbleToStake false"
101   @post (!stakeFlag || now < validStakeStartTime) -> __return == false
102
103 */
104 /*@CTK FAIL "isAbleToStake "
105   @pre stakeFlag == true
106   @pre now >= validStakeStartTime
107   @pre now == validStakeStartTime
108   @pre now == secondsInterval
109   @pre now - validStakeStartTime % secondsInterval < validStakeDuration
110   @post __return == false
111
112 */
113 function isAbleToStake() public view returns (bool){
114   if(!stakeFlag || now < validStakeStartTime) { //solium-disable-line
115     return false;
116   }
117   uint256 validTime = now.sub(validStakeStartTime).mod(secondsInterval); //solium-disable-
        line
118   return (validTime >= 0 && validTime < validStakeDuration);
119 }
120
121 /**
122 * @dev A method for check period for stake
123 */
124 function stakePeriod() public view returns (uint256 validStakePeriodStart, uint256
```

```
     validStakePeriodEnd){
125   if(now < validStakeStartTime) { //solium-disable-line
126     validStakePeriodStart = validStakeStartTime;
127     validStakePeriodEnd = validStakePeriodStart.add(validStakeDuration);
128   }else{
129     uint256 week = now.sub(validStakeStartTime).div(secondsInterval); //solium-disable-
          line
130
131     if(isAbleToStake()){
132       validStakePeriodStart = validStakeStartTime.add(secondsInterval.mul(week));
133     }else{
134       validStakePeriodStart = validStakeStartTime.add(secondsInterval.mul(week.add(1)));
135     }
136     validStakePeriodEnd = validStakePeriodStart.add(validStakeDuration);
137   }
138 }
139 // ---------- STAKES ----------
140 /**
141 * @dev A method for a stakeholder to create a stake.
142 * @param _stake The size of the stake to be created.
143 */
144 function createStake(uint256 _stake) external isvalidStakePeriod {
145   require(transfer(address(this), _stake), "Token transfer failed");
146   if(stakes[msg.sender] == 0) _addStakeholder(msg.sender);
147   stakes[msg.sender] = stakes[msg.sender].add(_stake);
148
149   emit Staked(msg.sender,_stake);
150 }
151
152 /**
153 * @dev A method for a stakeholder to remove a stake.
154 * @param _stake The size of the stake to be removed.
155 */
156 function removeStake(uint256 _stake) external isvalidStakePeriod {
157   require(_stake <= stakes[msg.sender], "insufficient stake balance");
158   stakes[msg.sender] = stakes[msg.sender].sub(_stake);
159   if(stakes[msg.sender] == 0) _removeStakeholder(msg.sender);
160
161   _transfer(address(this), msg.sender, _stake);
162   emit Unstaked(msg.sender,_stake);
163 }
164
165 /**
166 * @dev A method to retrieve the stake for a stakeholder.
167 * @param _stakeholder The stakeholder to retrieve the stake for.
168 * @return uint256 The amount of wei staked.
169 */
170 /*@CTK "stakeOf"
171   @post __return == stakes[_stakeholder]
172 */
173 function stakeOf(address _stakeholder) public view returns(uint256) {
174   return stakes[_stakeholder];
175 }
176
177 /**
178 * @dev A method to the aggregated stakes from all stakeholders.
179 * @return uint256 The aggregated stakes from all stakeholders.
180 */
```

```
181   function totalStakes() public view returns(uint256) {
182     uint256 _totalStakes = 0;
183     uint256 s = 0;
184     for ( s ; s < stakeholders.length; s += 1){
185       _totalStakes = _totalStakes.add(stakes[stakeholders[s]]);
186     }
187     return _totalStakes;
188   }
189
190   // ---------- STAKEHOLDERS ----------
191   /**
192   * @dev A method to check if an address is a stakeholder.
193   * @param _address The address to verify.
194   * @return bool, uint256 Whether the address is a stakeholder,
195   * and if so its position in the stakeholders array.
196   */
197   function isStakeholder(address _address) public view returns(bool, uint256) {
198     uint256 s = 0;
199     for ( s ; s < stakeholders.length; s += 1){
200       if (_address == stakeholders[s]) return (true, s);
201     }
202     return (false, 0);
203   }
204
205   /**
206   * @dev A method to add a stakeholder.
207   * @param _stakeholder The stakeholder to add.
208   */
209   function _addStakeholder(address _stakeholder) internal {
210     (bool _isStakeholder, ) = isStakeholder(_stakeholder);
211     if(!_isStakeholder) stakeholders.push(_stakeholder);
212   }
213
214   /**
215   * @dev A method to remove a stakeholder.
216   * @param _stakeholder The stakeholder to remove.
217   */
218   function _removeStakeholder(address _stakeholder) internal {
219     (bool _isStakeholder, uint256 s) = isStakeholder(_stakeholder);
220     if(_isStakeholder){
221       stakeholders[s] = stakeholders[stakeholders.length - 1];
222       stakeholders.pop();
223     }
224   }
225
226   /**
227   * @dev A method to get stake holders
228   */
229   /*@CTK stakeholderList
230     @post __return == stakeholders
231   */
232   function stakeholderList() external view returns (address[] memory) {
233     return stakeholders;
234   }
235 }
```

File token/LockableToken.sol

```
1 pragma solidity ^0.5.6;
```

```
2
3  import "./PausableToken.sol";
4  import "../erc/ERC1132.sol";
5
6  contract LockableToken is PausableToken, ERC1132 {
7
8
9    /**
10   * @dev Error messages for require statements
11   */
12   string internal constant ALREADY_LOCKED = "Tokens already locked";
13   string internal constant NOT_LOCKED = "No tokens locked";
14   string internal constant AMOUNT_ZERO = "Amount can not be 0";
15
16   /**
17   * @dev is msg.sender or shitelistadmin
18
19   * @param _address address
20   */
21   modifier isAdminOrSelf(address _address) {
22     require(_address == msg.sender || isWhitelistAdmin(msg.sender), "tokens are unlockable
         by owner or admin");
23     _;
24   }
25
26   // hack
27
28   mapping(address => bytes32[]) public lockReason;
29
30   mapping(address => mapping(bytes32 => LockToken)) public locked;
31   bool private _paused;
32
33   mapping (address => uint256) internal _balances;
34
35   Roles.Role private _whitelistAdmins;
36   //end hack
37
38   /**
39   * @dev Returns tokens locked for a specified address for a specified reason
40   * @param _of The address whose tokens are locked
41   * @param _reason The reason to query the lock tokens for
42   */
43   //@CTK NO_BUF_OVERFLOW
44   //@CTK NO_OVERFLOW
45   //@CTK NO_ASF
46   /*@CTK "tokensLocked"
47     @tag assume_completion
48     @post locked[_of][_reason].claimed -> amount == 0
49     @post !locked[_of][_reason].claimed -> amount == locked[_of][_reason].amount
50   */
51   function tokensLocked(address _of, bytes32 _reason)
52       public
53       view
54       returns (uint256 amount)
55   {
56     if (!locked[_of][_reason].claimed)
57         amount = locked[_of][_reason].amount;
58   }
```

```
59
60  /**
61  * @dev Returns tokens locked for a specified address for a
62  *     specified reason at a specific time
63  * @param _of The address whose tokens are locked
64  * @param _reason The reason to query the lock tokens for
65  * @param _time The timestamp to query the lock tokens for
66  */
67  //@CTK NO_BUF_OVERFLOW
68  //@CTK NO_OVERFLOW
69  //@CTK NO_ASF
70  /*@CTK "tokensLockedAtTime"
71    @tag assume_completion
72    @post locked[_of][_reason].validity <= _time -> amount == 0
73    @post locked[_of][_reason].validity > _time -> amount == locked[_of][_reason].amount
74  */
75  function tokensLockedAtTime(address _of, bytes32 _reason, uint256 _time)
76      public
77      view
78      returns (uint256 amount)
79  {
80    if (locked[_of][_reason].validity > _time)
81        amount = locked[_of][_reason].amount;
82  }
83
84  /**
85  * @dev Returns tokens validity for a specified address for a specified reason
86  *
87  * @param _of The address whose tokens are locked
88  * @param _reason The reason to query the lock tokens for
89  */
90  /*@CTK "tokensValidity"
91    @tag assume_completion
92    @post validity == locked[_of][_reason].validity
93  */
94  function tokensValidity(address _of, bytes32 _reason)
95      public
96      view
97      returns (uint256 validity)
98  {
99    validity = locked[_of][_reason].validity;
100 }
101
102 /**
103 * @dev Returns total tokens held by an address (locked + transferable)
104 * @param _of The address to query the total balance of
105 */
106 //@CTK NO_ASF
107 function lockBalanceOf(address _of)
108     public
109     view
110     returns (uint256 amount)
111 {
112   amount = 0;
113
114   uint256 i = 0;
115
116   /*@CTK getTotallockedBalance
```

```
117       @inv _of == _of__pre
118       @inv i <= this.lockReason[_of].length
119       @inv amount >= amount
120       @post i == this.lockReason[_of].length
121       @post !__should_return
122     */
123     for ( i ; i < lockReason[_of].length; i++) {
124       amount = amount.add(tokensLocked(_of, lockReason[_of][i]));
125     }
126   }
127
128   /**
129   * @dev Returns unlockable tokens for a specified address for a specified reason
130   * @param _of The address to query the the unlockable token count of
131   * @param _reason The reason to query the unlockable tokens for
132   */
133   /*@CTK tokensUnlockable
134     @tag assume_completion
135     @post (locked[_of][_reason].validity > now || locked[_of][_reason].claimed) -> amount ==
            0
136     @post (locked[_of][_reason].validity <= now && !locked[_of][_reason].claimed) -> amount
           == locked[_of][_reason].amount
137   */
138   function tokensUnlockable(address _of, bytes32 _reason)
139       public
140       view
141       returns (uint256 amount)
142   {
143       if (locked[_of][_reason].validity <= now && !locked[_of][_reason].claimed) //solium-
              disable-line
144           amount = locked[_of][_reason].amount;
145   }
146
147   /**
148   * @dev Unlocks the unlockable tokens of a specified address
149   * @param _of Address of user, claiming back unlockable tokens
150   */
151   //@CTK NO_BUF_OVERFLOW
152   //@CTK NO_OVERFLOW
153   //@CTK NO_ASF
154   function unlock(address _of)
155       public
156       isAdminOrSelf(_of)
157       returns (uint256 unlockableTokens)
158   {
159     uint256 lockedTokens;
160     uint256 i = 0;
161     /*#CTK "Forloop_unlockAll"
162       @inv i <= lockReason[_of].length
163       @inv forall j: uint. (j >= 0 /\ j < i /\ (locked[_of][lockReason[_of][i]].validity <=
              now && !locked[_of][lockReason[_of][i]].claimed && locked[_of][lockReason[_of][i]].
              amount > 0)) -> locked[_of][lockReason[_of][i]].claimed
164       @post i == lockReason[_of].length
165       @post !__should_return
166     */
167     /*@CTK loop
168       @inv true
169     */
```

```
170     for ( i ; i < lockReason[_of].length; i++) {
171       lockedTokens = tokensUnlockable(_of, lockReason[_of][i]);
172       if (lockedTokens > 0) {
173         unlockableTokens = unlockableTokens.add(lockedTokens);
174         locked[_of][lockReason[_of][i]].claimed = true;
175         emit Unlocked(_of, lockReason[_of][i], lockedTokens);
176       }
177     }
178     if (unlockableTokens > 0) {
179       _transfer(address(this), _of, unlockableTokens);
180     }
181   }
182
183   /**
184   * @dev Gets the unlockable tokens of a specified address
185   * @param _of The address to query the the unlockable token count of
186   */
187   //@CTK NO_BUF_OVERFLOW
188   //@CTK NO_OVERFLOW
189   //@CTK NO_ASF
190   function getUnlockableTokens(address _of)
191       public
192       view
193       returns (uint256 unlockableTokens)
194   {
195     uint256 i = 0;
196     /*@CTK Forloop_getUnlockableTokens
197       @inv _of == _of__pre
198       @inv i <= this.lockReason[_of].length
199       @inv unlockableTokens >= unlockableTokens
200       @post i == this.lockReason[_of].length
201       @post !__should_return
202     */
203     for ( i ; i < lockReason[_of].length; i++) {
204       unlockableTokens = unlockableTokens.add(tokensUnlockable(_of, lockReason[_of][i]));
205     }
206   }
207
208   /**
209   * @dev Transfers and Locks a specified amount of tokens,
210   *      for a specified reason and time
211   * @param _to adress to which tokens are to be transfered
212   * @param _reason The reason to lock tokens
213   * @param _amount Number of tokens to be transfered and locked
214   * @param _time Lock time in seconds
215   */
216   //@CTK NO_BUF_OVERFLOW
217   //@CTK NO_ASF
218   /*#CTK transferWithLock
219     @tag assume_completion
220     @post _amount == 0 -> __reverted
221     @post (locked[_to][_reason].claimed || locked[_to][_reason].amount == 0 ) -> __reverted
222     @post locked[msg.sender][_reason].amount == 0 ->
223       __post.lockReason[msg.sender].length == lockReason[msg.sender].length + 1
224     @post __return == true
225   */
226   function transferWithLock(address _to, bytes32 _reason, uint256 _amount, uint256 _time)
       public onlyWhitelistAdmin returns (bool) {
```

```
227     require(_to != address(0), "Zero address not allowed");
228     require(_amount != 0, AMOUNT_ZERO);
229     require(tokensLocked(_to, _reason) == 0, ALREADY_LOCKED);
230     require(balanceOf(msg.sender) > _amount);
231
232     uint256 validUntil = now.add(_time); //solium-disable-line
233
234     // not allowed duplicate reason for address
235     if (locked[_to][_reason].amount == 0)
236       lockReason[_to].push(_reason);
237
238     _transfer(msg.sender, address(this), _amount);
239
240     locked[_to][_reason] = LockToken(_amount, validUntil, false);
241
242     emit Locked(_to, _reason, _amount, validUntil);
243     return true;
244   }
245
246   /**
247   * @dev Extends lock for a specified reason and time
248   * @param _to adress to which tokens are to be extended lock
249   * @param _reason The reason to lock tokens
250   * @param _time Lock extension time in seconds
251   */
252   //@CTK NO_BUF_OVERFLOW
253   //@CTK NO_OVERFLOW
254   //@CTK NO_ASF
255   /*@CTK extendLock
256     @tag assume_completion
257     @pre (!locked[_to][_reason].claimed && locked[_to][_reason].amount > 0)
258     @post !_whitelistAdmins.bearer[msg.sender] -> __reverted
259     @post __post.locked[_to][_reason].validity == locked[_to][_reason].validity + _time
260   */
261   function extendLock(address _to, bytes32 _reason, uint256 _time) public onlyWhitelistAdmin
          returns (bool) {
262     require(_to != address(0), "Zero address not allowed");
263     require(tokensLocked(_to, _reason) > 0, NOT_LOCKED);
264
265     locked[_to][_reason].validity = locked[_to][_reason].validity.add(_time);
266
267     emit Locked(_to, _reason, locked[_to][_reason].amount, locked[_to][_reason].validity);
268     return true;
269   }
270
271   /**
272   * @dev Increase number of tokens locked for a specified reason
273   * @param _to adress to which tokens are to be increased
274   * @param _reason The reason to lock tokens
275   * @param _amount Number of tokens to be increased
276   */
277   //@CTK NO_BUF_OVERFLOW
278   //@CTK NO_OVERFLOW
279   //@CTK NO_ASF
280   /*@CTK increaseLockAmount
281     @tag assume_completion
282     @pre _to != msg.sender
283     @post !_whitelistAdmins.bearer[msg.sender] -> __reverted
```

```
284     @post (locked[_to][_reason].claimed || locked[_to][_reason].amount == 0 ) -> __reverted
285     @post (!locked[_to][_reason].claimed && locked[_to][_reason].amount > 0) ->
286       __post.locked[_to][_reason].amount == locked[_to][_reason].amount + _amount
287   */
288   function increaseLockAmount(address _to, bytes32 _reason, uint256 _amount) public
          onlyWhitelistAdmin returns (bool) {
289     require(_to != address(0), "Zero address not allowed");
290     require(tokensLocked(_to, _reason) > 0, NOT_LOCKED);
291     _transfer(msg.sender, address(this), _amount);
292
293     locked[_to][_reason].amount = locked[_to][_reason].amount.add(_amount);
294
295     emit Locked(_to, _reason, locked[_to][_reason].amount, locked[_to][_reason].validity);
296     return true;
297   }
298
299   /**
300   * @dev get length of lockReason array
301
302   * @param _of address
303   */
304   /*@CTK getSizeOfLockReason
305     @tag assume_completion
306     @post __return == lockReason[_of].length
307   */
308   function getSizeOfLockReason(address _of) public view returns (uint256) {
309     return lockReason[_of].length;
310   }
311 }
```

File roles/AdminRole.sol

```
1  pragma solidity ^0.5.6;
2
3  import "../@openzeppelin/contracts/access/roles/WhitelistAdminRole.sol";
4
5
6  contract AdminRole is WhitelistAdminRole {
7    //hack
8    // struct Role {
9    //   mapping (address => bool) bearer;
10   // }
11
12   //Roles.Role private _whitelistAdmins;
13   // end hack
14
15   address[] private _adminList;
16
17   /*@CTK _addWhitelistAdmin
18     @tag assume_completion
19     @post __post._adminList.length == _adminList.length + 1
20     @post __post._adminList[_adminList.length] == account
21   */
22   function _addWhitelistAdmin(address account) internal {
23     super._addWhitelistAdmin(account);
24     _adminList.push(account);
25   }
26
27   /*#CTK "_removeWhitelistAdmin"
```

```
28      @tag assume_completion
29      @post _adminList.length > 1 -> __post._adminList.length == _adminList.length - 1
30    */
31    function _removeWhitelistAdmin(address account) internal {
32      require(_adminList.length > 1, "At lease 1 Admin");
33
34      super._removeWhitelistAdmin(account);
35      uint256 s = getIndexOfAdmin(account);
36      _adminList[s] = _adminList[_adminList.length - 1];
37      _adminList.pop();
38    }
39
40    /*#CTK getIndexOfAdmin
41      @post (exists s: uint. (s < _adminList.length /\ _adminList[s] == account)) == __return
42     */
43    function getIndexOfAdmin(address account) public view returns(uint256) {
44
45      uint256 s = 0;
46      /*#CTK Forloop_getIndexOfAdmin
47        @var uint s
48        @inv this == this__pre
49        @inv account == account__pre
50        @inv s <= _adminList.length
51        @inv !__should_return -> (forall j: uint. (j >= 0 /\ j < s) -> _adminList[j] !=
              account)
52        @inv !__return__pre
53        @inv __should_return == __return
54        @post __should_return -> (s < _adminList.length /\ _adminList[s] == account)
55        @post !__should_return -> (forall j: uint. (j >= 0 /\ j < s) -> signers[j] != account)
56        @post !__should_return -> (__return == __return__pre)
57        @post !__should_return -> s
58       */
59      for (s; s < _adminList.length; s += 1) {
60        if (account == _adminList[s])
61          return s;
62      }
63      return 0;
64    }
65
66
67
68    function getAdminList() public view onlyWhitelistAdmin returns(address[] memory) {
69      return _adminList;
70    }
71  }
```