CertiK Audit Report For HintChain

Request Date: 2019-05-22
Revision Date: 2019-05-28
Platform Name: Ethereum

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 HintChain(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.

Exective Summary

This report has been prepared as product of the Smart Contract Audit request by HintChain. This audit was conducted to discover issues and vulnerabilities in the source code of HintChain's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis and Manual Review, a comprehensive examination has been performed. The auditing process pays special attention to the following considerations.

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

Vulnerability Classification

For every issues found, CertiK categorizes them into 3 buckets based on its risk level:

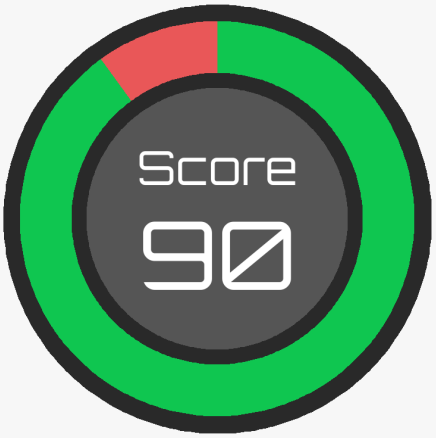
- Critical: The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.
- Medium: The code implementation does not match the specification at certain condition, or it could affect the security standard by lost of access control.
- Low: The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

Testing Summary

PASS

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

May 28, 2019



Type of Issues

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

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	1	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

tx.origin for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	4	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **VHC.sol** `ae49ba0c4fe807f6f96e8434203ad3fa7f1c1834b6b261eac0fa7ed0c188d0c8`

Summary

CertiK team is invited by The Hintchain team to audit the design and implementations of its to be released ERC20 based smart contract, and the source code has been analyzed under different perspectives and with different tools such as CertiK formal verification checking as well as manual reviews by smart contract experts. We have been actively interacting with client-side engineers when there was any potential loopholes or recommended design changes during the audit process, and Hintchain team has been actively giving us updates for the source code and feedback about the business logics.

At this point, the Hintchain team didn't provide other repositories sources as testing and documentation reference. We recommend having more unit tests coverage together with documentation to simulate potential use cases and walk through the functionalities to token holders, especially those super admin privileges that may impact the decentralized nature.

Overall we found the `VHC.sol` contract follows good practices, with a reasonable amount of features on top of the ERC20 related to administrative controls by the token issuer. With the final update of source code and delivery of the audit report, we conclude that the contract is 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 seeking multiple opinions, more test coverage, and sandbox deployments before the mainnet release.

Recommendations

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes.

VHC.sol/HINTToken

- **hintTransfer(), hintTransferFrom()** – Missing corresponding wrapping ERC-20 methods (`transfer`, `transferFrom`), as provided in `HINTBaseToken`.

VHC.sol/HINTBaseToken

- **hintBatchTransferToBounty()** – Missing length check for `userIdHash`. Also recommend consistent usage of `length` and `to.length`.
- **burnFrom()** – `burnFrom` in ERC-20 usually deducts caller's `allowance`. Renaming the current function as `burn` is more aligned with the common practices.
- **hintSell(), hintTransferToTeam(), hintTransferToPartner()** – Redundant address check `require(to != address(this), "...")` (already done in `hintTransferFrom`).

VHC.sol/DelayLockableToken

- **delayLockValues, delayLockBeforeValues, delayLockTimes**, – Recommend declaring these state variables as `internal`.
- **checkDelayUnlock()** – The amount locked in `super.lockValues` is not taken into account, which is inconsistent with the method `getMyUnlockValue()`.

VHC.sol/StandardToken

- **approve()** – Missing address check `spender != 0x0`.

VHC.sol/SafeMath

- **mul(), sub(), add()** – Recommend using `require` in place of `assert` for gas saving and error report.

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

Method will not encounter an assertion failure.

28, May 2019
37.41 ms

Line 24 in File VHC.sol

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

Line 31-42 in File VHC.sol

```
31    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
32        // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
33        // benefit is lost if 'b' is also tested.
34        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
35        if (a == 0) {
36            return 0;
37        }
38
39        c = a * b;
40        assert(c / a == b);
41        return c;
42    }
```

This code violates the specification

```
Counter Example:
Before Execution:
    Input = {
        a = 2
        b = 156
    }
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
        c = 0
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

```

```
33 Function invocation is reverted.
```

Formal Verification Request 2

SafeMath mul

28, May 2019
478.15 ms

Line 25-30 in File VHC.sol

```
25    /*@CTK "SafeMath mul"
26      @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
27      @post !__reverted -> c == a * b
28      @post !__reverted == !__has_overflow
29      @post !(__has_buf_overflow)
30     */
```

Line 31-42 in File VHC.sol

```
31    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
32        // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
33        // benefit is lost if 'b' is also tested.
34        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
35        if (a == 0) {
36            return 0;
37        }
38
39        c = a * b;
40        assert(c / a == b);
41        return c;
42    }
```

The code meets the specification

Formal Verification Request 3

SafeMath div

28, May 2019
8.48 ms

Line 47-53 in File VHC.sol

```
47    /*@CTK "SafeMath div"
48      @post b != 0 -> !__reverted
49      @post !__reverted -> __return == a / b
50      @post !__reverted -> !__has_overflow
51      @post !(__has_buf_overflow)
52      @post !__reverted -> !(__has_assertion_failure)
53    */
```

Line 54-59 in File VHC.sol

```
54    function div(uint256 a, uint256 b) internal pure returns (uint256) {
55        // assert(b > 0); // Solidity automatically throws when dividing by 0
56        // uint256 c = a / b;
```

```
57          // assert(a == b * c + a % b); // There is no case in which this doesn't hold
58          return a / b;
59      }
```

The code meets the specification

Formal Verification Request 4

Method will not encounter an assertion failure.

28, May 2019
20.32 ms

Line 64 in File VHC.sol

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

Line 71-74 in File VHC.sol

```
71      function sub(uint256 a, uint256 b) internal pure returns (uint256) {
72          assert(b <= a);
73          return a - b;
74      }
```

This code violates the specification

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

Function invocation is reverted.
```

Formal Verification Request 5

SafeMath sub

28, May 2019
2.05 ms

Line 65-70 in File VHC.sol

```
65    /*@CTK "SafeMath sub"
66      @post (a < b) == __reverted
67      @post !__reverted -> __return == a - b
68      @post !__reverted -> !__has_overflow
69      @post !(__has_buf_overflow)
70    */
```

Line 71-74 in File VHC.sol

```
71    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
72        assert(b <= a);
73        return a - b;
74    }
```

The code meets the specification

Formal Verification Request 6

Method will not encounter an assertion failure.

28, May 2019
19.12 ms

Line 79 in File VHC.sol

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

Line 86-90 in File VHC.sol

```
86    function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
87        c = a + b;
88        assert(c >= a);
89        return c;
90    }
```

This code violates the specification

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          a = 143
5          b = 113
6      }
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
```

```
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        block = {
          "number": 0,
          "timestamp": 0
        }
        c = 0
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 7

SafeMath add

28, May 2019
4.51 ms

Line 80-85 in File VHC.sol

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

Line 86-90 in File VHC.sol

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

The code meets the specification

Formal Verification Request 8

totalSupply correctness

28, May 2019
6.86 ms

Line 107-109 in File VHC.sol

```
107     /*@CTK "totalSupply correctness"
108       @post __return == totalSupply_
109      */
```

Line 110-112 in File VHC.sol

```
110     function totalSupply() public view returns (uint256) {
111         return totalSupply_;
112     }
```

The code meets the specification

Formal Verification Request 9

balanceOf correctness

28, May 2019
6.21 ms

Line 134-136 in File VHC.sol

```
134     /*@CTK "balanceOf correctness"
135       @post __return == balances[_owner]
136      */
```

Line 137-139 in File VHC.sol

```
137     function balanceOf(address _owner) public view returns (uint256) {
138         return balances[_owner];
139     }
```

The code meets the specification

Formal Verification Request 10

changeRoot correctness

28, May 2019
49.98 ms

Line 358-365 in File VHC.sol

```
358     /*@CTK "changeRoot correctness"
359       @tag assume_completion
360       @post !(__has_overflow)
361       @post !(__has_buf_overflow)
362       @post !(__has_assertion_failure)
363       @post newRoot != 0x0
364       @post __post.root == newRoot
365      */
```

Line 366-373 in File VHC.sol

```
366     function changeRoot(address newRoot) onlyRoot public returns (bool) {
367         require(newRoot != address(0), "This address to be set is zero address(0).
                Check the input address.");
368
369         root = newRoot;
```

```
370
371         emit ChangedRoot(newRoot);
372         return true;
373     }
```

The code meets the specification

Formal Verification Request 11

changeSuperOwner correctness

28, May 2019
49.6 ms

Line 379-387 in File VHC.sol

```
379     /*@CTK "changeSuperOwner correctness"
380       @tag assume_completion
381       @post !(__has_overflow)
382       @post !(__has_buf_overflow)
383       @post !(__has_assertion_failure)
384       @post msg.sender == root
385       @post newSuperOwner != 0x0
386       @post __post.superOwner == newSuperOwner
387      */
```

Line 388-395 in File VHC.sol

```
388     function changeSuperOwner(address newSuperOwner) onlyRoot public returns (bool) {
389         require(newSuperOwner != address(0), "This address to be set is zero address(0)
                . Check the input address.");
390
391         superOwner = newSuperOwner;
392
393         emit ChangedSuperOwner(newSuperOwner);
394         return true;
395     }
```

The code meets the specification

Formal Verification Request 12

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

28, May 2019
70.69 ms

Line 428 in File VHC.sol

```
428     //@CTK NO_BUF_OVERFLOW
```

Line 439-448 in File VHC.sol

```
439     function newOwner(address owner) onlySuperOwner public returns (bool) {
440         require(owner != address(0), "This address to be set is zero address(0). Check
                the input address.");
441         require(!owners[owner], "This address is already registered.");
```

```
442
443         owners[owner] = true;
444         ownerList.push(owner);
445
446         emit AddedNewOwner(owner);
447         return true;
448     }
```

The code meets the specification

Formal Verification Request 13

Method will not encounter an assertion failure.

28, May 2019
3.4 ms

Line 429 in File VHC.sol

```
429     //@CTK NO_ASF
```

Line 439-448 in File VHC.sol

```
439     function newOwner(address owner) onlySuperOwner public returns (bool) {
440         require(owner != address(0), "This address to be set is zero address(0). Check
                the input address.");
441         require(!owners[owner], "This address is already registered.");
442
443         owners[owner] = true;
444         ownerList.push(owner);
445
446         emit AddedNewOwner(owner);
447         return true;
448     }
```

The code meets the specification

Formal Verification Request 14

newOwner correctness

28, May 2019
19.79 ms

Line 430-438 in File VHC.sol

```
430     /*@CTK "newOwner correctness"
431       @tag assume_completion
432       @post msg.sender == superOwner
433       @post owner != 0x0
434       @post owners[owner] == false
435       @post __post.ownerList.length == ownerList.length + 1
436       @post __post.ownerList[ownerList.length] == owner
437       @post __post.owners[owner] == true
438      */
```

Line 439-448 in File VHC.sol

```
439     function newOwner(address owner) onlySuperOwner public returns (bool) {
440         require(owner != address(0), "This address to be set is zero address(0). Check
                the input address.");
441         require(!owners[owner], "This address is already registered.");
442 
443         owners[owner] = true;
444         ownerList.push(owner);
445 
446         emit AddedNewOwner(owner);
447         return true;
448     }
```

✅ The code meets the specification

Formal Verification Request 15

lock correctness

📅 28, May 2019
⏱ 29.37 ms

Line 501-508 in File VHC.sol

```
501     /*@CTK "lock correctness"
502       @tag assume_completion
503       @post !(__has_overflow)
504       @post !(__has_buf_overflow)
505       @post !(__has_assertion_failure)
506       @post owners[msg.sender] == true
507       @post __post.locked == true
508      */
```

Line 509-512 in File VHC.sol

```
509     function lock(string note) onlyOwner public {
510         locked = true;
511         emit Locked(locked, note);
512     }
```

✅ The code meets the specification

Formal Verification Request 16

unlock correctness

📅 28, May 2019
⏱ 28.51 ms

Line 514-521 in File VHC.sol

```
514     /*@CTK "unlock correctness"
515       @tag assume_completion
516       @post !(__has_overflow)
517       @post !(__has_buf_overflow)
518       @post !(__has_assertion_failure)
519       @post owners[msg.sender] == true
```

```
520       @post __post.locked == false
521     */
```

Line 522-525 in File VHC.sol

```
522     function unlock(string note) onlyOwner public {
523         locked = false;
524         emit Locked(locked, note);
525     }
```

The code meets the specification

Formal Verification Request 17

lockTo correctness

28, May 2019
90.36 ms

Line 527-536 in File VHC.sol

```
527     /*@CTK "lockTo correctness"
528       @tag assume_completion
529       @pre LOCK_MAX == 255
530       @post !(__has_overflow)
531       @post !(__has_buf_overflow)
532       @post !(__has_assertion_failure)
533       @post owners[msg.sender] == true
534       @post __post.lockValues[addr] == 255
535       @post __post.unlockAddrs[addr] == false
536     */
```

Line 537-542 in File VHC.sol

```
537     function lockTo(address addr, string note) onlyOwner public {
538         setLockValue(addr, LOCK_MAX, note);
539         unlockAddrs[addr] = false;
540 
541         emit LockedTo(addr, true, note);
542     }
```

The code meets the specification

Formal Verification Request 18

unlockTo correctness

28, May 2019
74.61 ms

Line 544-553 in File VHC.sol

```
544     /*@CTK "unlockTo correctness"
545       @tag assume_completion
546       @pre LOCK_MAX == 255
547       @post !(__has_overflow)
548       @post !(__has_buf_overflow)
```

```
549     @post !(__has_assertion_failure)
550     @post owners[msg.sender] == true
551     @post lockValues[addr] == 255 -> __post.lockValues[addr] == 0
552     @post __post.unlockAddrs[addr] == true
553    */
```

Line 554-560 in File VHC.sol

```
554   function unlockTo(address addr, string note) onlyOwner public {
555       if (lockValues[addr] == LOCK_MAX)
556           setLockValue(addr, 0, note);
557       unlockAddrs[addr] = true;
558
559       emit LockedTo(addr, false, note);
560   }
```

✅ The code meets the specification

Formal Verification Request 19

setLockValue correctness

📅 28, May 2019
⏱ 2.38 ms

Line 562-569 in File VHC.sol

```
562   /*@CTK "setLockValue correctness"
563     @tag assume_completion
564     @post !(__has_overflow)
565     @post !(__has_buf_overflow)
566     @post !(__has_assertion_failure)
567     @post owners[msg.sender] == true
568     @post __post.lockValues[addr] == value
569    */
```

Line 570-573 in File VHC.sol

```
570   function setLockValue(address addr, uint256 value, string note) onlyOwner public {
571       lockValues[addr] = value;
572       emit SetLockValue(addr, value, note);
573   }
```

✅ The code meets the specification

Formal Verification Request 20

getMyUnlockValue correctness

📅 28, May 2019
⏱ 74.68 ms

Line 578-593 in File VHC.sol

```
578   /*@CTK "getMyUnlockValue correctness"
579     @tag assume_completion
580     @post !(__has_overflow)
```

```
581      @post !(__has_buf_overflow)
582      @post !(__has_assertion_failure)
583      @post locked && !unlockAddrs[msg.sender]
584           -> __return == 0
585      @post !locked && (balances[msg.sender] > lockValues[msg.sender])
586           -> __return == balances[msg.sender] - lockValues[msg.sender]
587      @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] > lockValues[
             msg.sender])
588           -> __return == balances[msg.sender] - lockValues[msg.sender]
589      @post !locked && (balances[msg.sender] <= lockValues[msg.sender])
590           -> __return == 0
591      @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] <= lockValues[
             msg.sender])
592           -> __return == 0
593     */
```

Line 594-600 in File VHC.sol

```
594   function getMyUnlockValue() public view returns (uint256) {
595       address addr = msg.sender;
596       if ((!locked || unlockAddrs[addr]) && balances[addr] > lockValues[addr])
597           return balances[addr].sub(lockValues[addr]);
598       else
599           return 0;
600   }
```

The code meets the specification

Formal Verification Request 21

If method completes, integer overflow would not happen.

28, May 2019
198.26 ms

Line 635 in File VHC.sol

```
635   //@CTK FAIL NO_OVERFLOW
```

Line 645-660 in File VHC.sol

```
645   function delayLock(uint256 value) public returns (bool) {
646       require (value <= balances[msg.sender], "Your balance is insufficient.");
647
648       if (value >= delayLockValues[msg.sender])
649           delayLockTimes[msg.sender] = now;
650       else {
651           require (delayLockTimes[msg.sender] <= now, "The remaining money in the
                  account cannot be unlocked continuously. You cannot renew until 12
                  hours after the first run.");
652           delayLockTimes[msg.sender] = now + 12 hours;
653           delayLockBeforeValues[msg.sender] = delayLockValues[msg.sender];
654       }
655
656       delayLockValues[msg.sender] = value;
657
658       emit SetDelayLockValue(msg.sender, value, delayLockTimes[msg.sender]);
659       return true;
660   }
```

This code violates the specification

```
Counter Example:
Before Execution:
    Input = {
        value = 0
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = false
        __reverted = false
        msg = {
          "gas": 0,
          "sender": 0,
          "value": 0
        }
    }
    Other = {
        __return = false
        block = {
          "number": 0,
          "timestamp": 128
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "DelayLockableToken",
          "balance": 0,
          "contract": {
            "delayLockValues": [
              {
                "key": 66,
                "value": 2
              },
              {
                "key": 160,
                "value": 64
              },
              {
                "key": 4,
                "value": 128
              },
              {
                "key": 0,
                "value": 1
              },
              {
                "key": 132,
                "value": 32
              },
              {
                "key": 16,
                "value": 128
              },
```

```
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "delayLockBeforeValues": [
            {
              "key": 0,
              "value": 5
            },
            {
              "key": 144,
              "value": 16
            },
            {
              "key": 2,
              "value": 1
            },
            {
              "key": 68,
              "value": 32
            },
            {
              "key": 64,
              "value": 1
            },
            {
              "key": 4,
              "value": 4
            },
            {
              "key": 132,
              "value": 64
            },
            {
              "key": 8,
              "value": 24
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "delayLockTimes": [
            {
              "key": 32,
              "value": 2
            },
            {
              "key": 128,
              "value": 2
            },
            {
              "key": 36,
              "value": 64
            },
            {
              "key": 0,
```

```
            "value": 1
          },
          {
            "key": 16,
            "value": 64
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "locked": false,
        "LOCK_MAX": 0,
        "unlockAddrs": [
          {
            "key": 0,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "lockValues": [
          {
            "key": 33,
            "value": 8
          },
          {
            "key": 128,
            "value": 64
          },
          {
            "key": 192,
            "value": 32
          },
          {
            "key": 4,
            "value": 2
          },
          {
            "key": 0,
            "value": 16
          },
          {
            "key": 1,
            "value": 4
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "root": 0,
        "superOwner": 0,
        "owners": [
          {
            "key": "ALL_OTHERS",
```

```
            "value": false
          }
        ],
        "ownerList": [],
        "candidateSuperOwnerMap": [
          {
            "key": 32,
            "value": 8
          },
          {
            "key": 2,
            "value": 128
          },
          {
            "key": 0,
            "value": 2
          },
          {
            "key": 1,
            "value": 1
          },
          {
            "key": 8,
            "value": 64
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "allowed": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": 0
              }
            ]
          }
        ],
        "balances": [
          {
            "key": 24,
            "value": 32
          },
          {
            "key": 8,
            "value": 64
          },
          {
            "key": 128,
            "value": 16
          },
          {
            "key": 16,
            "value": 64
          },
```

```
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  Input = {
    value = 0
  }
  This = 0
  Internal = {
    __has_assertion_failure = false
    __has_buf_overflow = false
    __has_overflow = true
    __has_returned = true
    __reverted = false
    msg = {
      "gas": 0,
      "sender": 0,
      "value": 0
    }
  }
  Other = {
    __return = true
    block = {
      "number": 0,
      "timestamp": 128
    }
  }
  Address_Map = [
    {
      "key": 0,
      "value": {
        "contract_name": "DelayLockableToken",
        "balance": 0,
        "contract": {
          "delayLockValues": [
            {
              "key": 66,
              "value": 2
            },
            {
              "key": 160,
              "value": 64
            },
            {
              "key": 4,
              "value": 128
```

```
            },
            {
              "key": 132,
              "value": 32
            },
            {
              "key": 16,
              "value": 128
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "delayLockBeforeValues": [
            {
              "key": 144,
              "value": 16
            },
            {
              "key": 0,
              "value": 1
            },
            {
              "key": 2,
              "value": 1
            },
            {
              "key": 68,
              "value": 32
            },
            {
              "key": 64,
              "value": 1
            },
            {
              "key": 4,
              "value": 4
            },
            {
              "key": 132,
              "value": 64
            },
            {
              "key": 8,
              "value": 24
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "delayLockTimes": [
            {
              "key": 32,
              "value": 2
            },
            {
```

```
            "key": 128,
            "value": 2
          },
          {
            "key": 36,
            "value": 64
          },
          {
            "key": 0,
            "value": 64
          },
          {
            "key": 16,
            "value": 64
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "locked": false,
        "LOCK_MAX": 0,
        "unlockAddrs": [
          {
            "key": 0,
            "value": true
          },
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "lockValues": [
          {
            "key": 33,
            "value": 8
          },
          {
            "key": 128,
            "value": 64
          },
          {
            "key": 192,
            "value": 32
          },
          {
            "key": 4,
            "value": 2
          },
          {
            "key": 0,
            "value": 16
          },
          {
            "key": 1,
            "value": 4
          },
          {
```

```
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "root": 0,
        "superOwner": 0,
        "owners": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "ownerList": [],
        "candidateSuperOwnerMap": [
          {
            "key": 32,
            "value": 8
          },
          {
            "key": 2,
            "value": 128
          },
          {
            "key": 0,
            "value": 2
          },
          {
            "key": 1,
            "value": 1
          },
          {
            "key": 8,
            "value": 64
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "allowed": [
          {
            "key": "ALL_OTHERS",
            "value": [
              {
                "key": "ALL_OTHERS",
                "value": 0
              }
            ]
          }
        ],
        "balances": [
          {
            "key": 24,
            "value": 32
          },
          {
            "key": 8,
            "value": 64
```

```
            },
            {
              "key": 128,
              "value": 16
            },
            {
              "key": 16,
              "value": 64
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

Formal Verification Request 22

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

28, May 2019
1.19 ms

Line 636 in File VHC.sol

```
    //@CTK NO_BUF_OVERFLOW
```

Line 645-660 in File VHC.sol

```
    function delayLock(uint256 value) public returns (bool) {
        require (value <= balances[msg.sender], "Your balance is insufficient.");

        if (value >= delayLockValues[msg.sender])
            delayLockTimes[msg.sender] = now;
        else {
            require (delayLockTimes[msg.sender] <= now, "The remaining money in the
                account cannot be unlocked continuously. You cannot renew until 12
                hours after the first run.");
            delayLockTimes[msg.sender] = now + 12 hours;
            delayLockBeforeValues[msg.sender] = delayLockValues[msg.sender];
        }

        delayLockValues[msg.sender] = value;

        emit SetDelayLockValue(msg.sender, value, delayLockTimes[msg.sender]);
        return true;
    }
```

The code meets the specification

Formal Verification Request 23

Method will not encounter an assertion failure.

28, May 2019
1.06 ms

Line 637 in File VHC.sol

```
637     //@CTK NO_ASF
```

Line 645-660 in File VHC.sol

```
645     function delayLock(uint256 value) public returns (bool) {
646         require (value <= balances[msg.sender], "Your balance is insufficient.");
647 
648         if (value >= delayLockValues[msg.sender])
649             delayLockTimes[msg.sender] = now;
650         else {
651             require (delayLockTimes[msg.sender] <= now, "The remaining money in the
                    account cannot be unlocked continuously. You cannot renew until 12
                    hours after the first run.");
652             delayLockTimes[msg.sender] = now + 12 hours;
653             delayLockBeforeValues[msg.sender] = delayLockValues[msg.sender];
654         }
655 
656         delayLockValues[msg.sender] = value;
657 
658         emit SetDelayLockValue(msg.sender, value, delayLockTimes[msg.sender]);
659         return true;
660     }
```

The code meets the specification

Formal Verification Request 24

delayLock correctness

28, May 2019
42.68 ms

Line 638-644 in File VHC.sol

```
638     /*@CTK "delayLock correctness"
639       @tag assume_completion
640       @post value <= balances[msg.sender]
641       @post value < delayLockValues[msg.sender]
642             -> __post.delayLockBeforeValues[msg.sender] == delayLockValues[msg.sender]
643       @post __post.delayLockValues[msg.sender] == value
644      */
```

Line 645-660 in File VHC.sol

```
645     function delayLock(uint256 value) public returns (bool) {
646         require (value <= balances[msg.sender], "Your balance is insufficient.");
647 
648         if (value >= delayLockValues[msg.sender])
649             delayLockTimes[msg.sender] = now;
650         else {
```

```
651             require (delayLockTimes[msg.sender] <= now, "The remaining money in the
                    account cannot be unlocked continuously. You cannot renew until 12
                    hours after the first run.");
652             delayLockTimes[msg.sender] = now + 12 hours;
653             delayLockBeforeValues[msg.sender] = delayLockValues[msg.sender];
654         }
655
656         delayLockValues[msg.sender] = value;
657
658         emit SetDelayLockValue(msg.sender, value, delayLockTimes[msg.sender]);
659         return true;
660     }
```

✅ The code meets the specification

Formal Verification Request 25

If method completes, integer overflow would not happen.

📅 28, May 2019
⏱ 189.32 ms

Line 665 in File VHC.sol

```
665     //@CTK FAIL NO_OVERFLOW
```

Line 674-676 in File VHC.sol

```
674     function delayUnlock() public returns (bool) {
675         return delayLock(0);
676     }
```

❌ 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         __return = false
18         block = {
19           "number": 0,
20           "timestamp": 128
21         }
22     }
23     Address_Map = [
24       {
```

```
      "key": 0,
      "value": {
        "contract_name": "DelayLockableToken",
        "balance": 0,
        "contract": {
          "delayLockValues": [
            {
              "key": 1,
              "value": 2
            },
            {
              "key": 2,
              "value": 8
            },
            {
              "key": 4,
              "value": 2
            },
            {
              "key": 0,
              "value": 128
            },
            {
              "key": 8,
              "value": 8
            },
            {
              "key": 128,
              "value": 4
            },
            {
              "key": 16,
              "value": 32
            },
            {
              "key": 64,
              "value": 8
            },
            {
              "key": 68,
              "value": 16
            },
            {
              "key": 20,
              "value": 16
            },
            {
              "key": 18,
              "value": 4
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "delayLockBeforeValues": [
            {
              "key": 1,
```

```
            "value": 64
          },
          {
            "key": 0,
            "value": 130
          },
          {
            "key": 32,
            "value": 2
          },
          {
            "key": 16,
            "value": 4
          },
          {
            "key": 33,
            "value": 8
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "delayLockTimes": [
          {
            "key": 4,
            "value": 64
          },
          {
            "key": 0,
            "value": 1
          },
          {
            "key": 16,
            "value": 8
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "locked": false,
        "LOCK_MAX": 0,
        "unlockAddrs": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "lockValues": [
          {
            "key": 4,
            "value": 1
          },
          {
            "key": 0,
            "value": 16
          },
```

```
            {
              "key": 8,
              "value": 1
            },
            {
              "key": 128,
              "value": 2
            },
            {
              "key": 136,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "root": 0,
          "superOwner": 0,
          "owners": [
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "ownerList": [],
          "candidateSuperOwnerMap": [
            {
              "key": 2,
              "value": 32
            },
            {
              "key": 0,
              "value": 5
            },
            {
              "key": 8,
              "value": 4
            },
            {
              "key": 128,
              "value": 2
            },
            {
              "key": 16,
              "value": 32
            },
            {
              "key": 10,
              "value": 128
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
```

```
            }
          ],
          "allowed": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 0
                }
              ]
            }
          ],
          "balances": [
            {
              "key": 1,
              "value": 64
            },
            {
              "key": 2,
              "value": 16
            },
            {
              "key": 32,
              "value": 32
            },
            {
              "key": 128,
              "value": 4
            },
            {
              "key": 64,
              "value": 4
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
  This = 0
  Internal = {
    __has_assertion_failure = false
    __has_buf_overflow = false
    __has_overflow = true
    __has_returned = true
    __reverted = false
    msg = {
```

```
        "gas": 0,
        "sender": 0,
        "value": 0
      }
    }
    Other = {
      __return = true
      block = {
        "number": 0,
        "timestamp": 128
      }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "DelayLockableToken",
          "balance": 0,
          "contract": {
            "delayLockValues": [
              {
                "key": 1,
                "value": 2
              },
              {
                "key": 18,
                "value": 4
              },
              {
                "key": 4,
                "value": 2
              },
              {
                "key": 8,
                "value": 8
              },
              {
                "key": 128,
                "value": 4
              },
              {
                "key": 16,
                "value": 32
              },
              {
                "key": 64,
                "value": 8
              },
              {
                "key": 68,
                "value": 16
              },
              {
                "key": 20,
                "value": 16
              },
              {
                "key": 2,
```

```
            "value": 8
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "delayLockBeforeValues": [
          {
            "key": 33,
            "value": 8
          },
          {
            "key": 0,
            "value": 128
          },
          {
            "key": 32,
            "value": 2
          },
          {
            "key": 16,
            "value": 4
          },
          {
            "key": 1,
            "value": 64
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "delayLockTimes": [
          {
            "key": 4,
            "value": 64
          },
          {
            "key": 0,
            "value": 64
          },
          {
            "key": 16,
            "value": 8
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "locked": false,
        "LOCK_MAX": 0,
        "unlockAddrs": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
```

```
          ],
          "lockValues": [
            {
              "key": 4,
              "value": 1
            },
            {
              "key": 0,
              "value": 16
            },
            {
              "key": 8,
              "value": 1
            },
            {
              "key": 128,
              "value": 2
            },
            {
              "key": 136,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "root": 0,
          "superOwner": 0,
          "owners": [
            {
              "key": 0,
              "value": true
            },
            {
              "key": "ALL_OTHERS",
              "value": false
            }
          ],
          "ownerList": [],
          "candidateSuperOwnerMap": [
            {
              "key": 2,
              "value": 32
            },
            {
              "key": 0,
              "value": 5
            },
            {
              "key": 8,
              "value": 4
            },
            {
              "key": 128,
              "value": 2
            },
            {
```

```
              "key": 16,
              "value": 32
            },
            {
              "key": 10,
              "value": 128
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "allowed": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 0
                }
              ]
            }
          ],
          "balances": [
            {
              "key": 1,
              "value": 64
            },
            {
              "key": 2,
              "value": 16
            },
            {
              "key": 32,
              "value": 32
            },
            {
              "key": 128,
              "value": 4
            },
            {
              "key": 64,
              "value": 4
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

Formal Verification Request 26

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

28, May 2019
1.24 ms

Line 666 in File VHC.sol

```
666     //@CTK NO_BUF_OVERFLOW
```

Line 674-676 in File VHC.sol

```
674     function delayUnlock() public returns (bool) {
675         return delayLock(0);
676     }
```

The code meets the specification

Formal Verification Request 27

Method will not encounter an assertion failure.

28, May 2019
1.42 ms

Line 667 in File VHC.sol

```
667     //@CTK NO_ASF
```

Line 674-676 in File VHC.sol

```
674     function delayUnlock() public returns (bool) {
675         return delayLock(0);
676     }
```

The code meets the specification

Formal Verification Request 28

delayUnlock correctness

28, May 2019
38.84 ms

Line 668-673 in File VHC.sol

```
668     /*@CTK "delayUnlock correctness"
669       @tag assume_completion
670       @post delayLockValues[msg.sender] != 0
671             -> __post.delayLockBeforeValues[msg.sender] == delayLockValues[msg.sender]
672       @post __post.delayLockValues[msg.sender] == 0
673      */
```

Line 674-676 in File VHC.sol

```
674     function delayUnlock() public returns (bool) {
675         return delayLock(0);
676     }
```

The code meets the specification

Formal Verification Request 29

If method completes, integer overflow would not happen.

28, May 2019
237.7 ms

Line 681 in File VHC.sol

```
681     //@CTK NO_OVERFLOW
```

Line 711-726 in File VHC.sol

```
711     function getMyUnlockValue() public view returns (uint256) {
712         uint256 myUnlockValue;
713         address addr = msg.sender;
714         if (delayLockTimes[addr] <= now) {
715             myUnlockValue = balances[addr].sub(delayLockValues[addr]);
716         } else {
717             myUnlockValue = balances[addr].sub(delayLockBeforeValues[addr]);
718         }
719 
720         uint256 superUnlockValue = super.getMyUnlockValue();
721 
722         if (myUnlockValue > superUnlockValue)
723             return superUnlockValue;
724         else
725             return myUnlockValue;
726     }
```

The code meets the specification

Formal Verification Request 30

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

28, May 2019
25.36 ms

Line 682 in File VHC.sol

```
682     //@CTK NO_BUF_OVERFLOW
```

Line 711-726 in File VHC.sol

```
711     function getMyUnlockValue() public view returns (uint256) {
712         uint256 myUnlockValue;
713         address addr = msg.sender;
714         if (delayLockTimes[addr] <= now) {
715             myUnlockValue = balances[addr].sub(delayLockValues[addr]);
716         } else {
717             myUnlockValue = balances[addr].sub(delayLockBeforeValues[addr]);
718         }
719 
720         uint256 superUnlockValue = super.getMyUnlockValue();
```

```
721
722         if (myUnlockValue > superUnlockValue)
723             return superUnlockValue;
724         else
725             return myUnlockValue;
726     }
```

The code meets the specification

Formal Verification Request 31

Method will not encounter an assertion failure.

28, May 2019
92.27 ms

Line 683 in File VHC.sol

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

Line 711-726 in File VHC.sol

```
711     function getMyUnlockValue() public view returns (uint256) {
712         uint256 myUnlockValue;
713         address addr = msg.sender;
714         if (delayLockTimes[addr] <= now) {
715             myUnlockValue = balances[addr].sub(delayLockValues[addr]);
716         } else {
717             myUnlockValue = balances[addr].sub(delayLockBeforeValues[addr]);
718         }
719
720         uint256 superUnlockValue = super.getMyUnlockValue();
721
722         if (myUnlockValue > superUnlockValue)
723             return superUnlockValue;
724         else
725             return myUnlockValue;
726     }
```

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         __return = 0
```

```
      block = {
        "number": 0,
        "timestamp": 128
      }
  }
  Address_Map = [
    {
      "key": "ALL_OTHERS",
      "value": {
        "contract_name": "DelayLockableToken",
        "balance": 0,
        "contract": {
          "delayLockValues": [
            {
              "key": 64,
              "value": 128
            },
            {
              "key": 0,
              "value": 9
            },
            {
              "key": 32,
              "value": 128
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "delayLockBeforeValues": [
            {
              "key": 128,
              "value": 1
            },
            {
              "key": 2,
              "value": 4
            },
            {
              "key": 0,
              "value": 8
            },
            {
              "key": 8,
              "value": 32
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "delayLockTimes": [
            {
              "key": 0,
              "value": 64
            },
            {
```

```
            "key": 16,
            "value": 8
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "locked": false,
        "LOCK_MAX": 0,
        "unlockAddrs": [
          {
            "key": "ALL_OTHERS",
            "value": false
          }
        ],
        "lockValues": [
          {
            "key": 128,
            "value": 2
          },
          {
            "key": 32,
            "value": 64
          },
          {
            "key": "ALL_OTHERS",
            "value": 0
          }
        ],
        "root": 0,
        "superOwner": 0,
        "owners": [
          {
            "key": "ALL_OTHERS",
            "value": true
          }
        ],
        "ownerList": [],
        "candidateSuperOwnerMap": [
          {
            "key": 128,
            "value": 64
          },
          {
            "key": 0,
            "value": 8
          },
          {
            "key": 1,
            "value": 128
          },
          {
            "key": 4,
            "value": 4
          },
          {
            "key": 16,
```

```
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 32
            }
          ],
          "allowed": [
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": 64,
                  "value": 128
                },
                {
                  "key": 0,
                  "value": 9
                },
                {
                  "key": 32,
                  "value": 128
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 0
                }
              ]
            }
          ],
          "balances": [
            {
              "key": 2,
              "value": 2
            },
            {
              "key": 16,
              "value": 1
            },
            {
              "key": "ALL_OTHERS",
              "value": 0
            }
          ],
          "totalSupply_": 0
        }
      }
    }
  ]

Function invocation is reverted.
```


Formal Verification Request 32

DelayLockableToken getMyUnlockValue correctness

📅 28, May 2019

4818.77 ms

Line 684-710 in File VHC.sol

```
684     /*@CTK "DelayLockableToken getMyUnlockValue correctness"
685       @tag assume_completion
686       @pre balances[msg.sender] >= delayLockValues[msg.sender]
687       @pre balances[msg.sender] >= delayLockBeforeValues[msg.sender]
688       @post locked && !unlockAddrs[msg.sender]
689            -> __return == 0
690       @post !locked && (balances[msg.sender] <= lockValues[msg.sender])
691            -> __return == 0
692       @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] <= lockValues[
              msg.sender])
693            -> __return == 0
694       @post !locked && (balances[msg.sender] > lockValues[msg.sender])
695            && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                 delayLockValues[msg.sender]
696            && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                 delayLockBeforeValues[msg.sender]
697            -> __return == balances[msg.sender] - lockValues[msg.sender]
698       @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] > lockValues[
              msg.sender])
699            && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                 delayLockValues[msg.sender]
700            && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                 delayLockBeforeValues[msg.sender]
701            -> __return == balances[msg.sender] - lockValues[msg.sender]
702       @post !locked && (balances[msg.sender] > lockValues[msg.sender])
703            && balances[msg.sender] - lockValues[msg.sender] >= balances[msg.sender] -
                 delayLockValues[msg.sender]
704            && delayLockTimes[msg.sender] <= now
705            -> __return == balances[msg.sender] - delayLockValues[msg.sender]
706       @post !locked && (balances[msg.sender] > lockValues[msg.sender])
707            && balances[msg.sender] - lockValues[msg.sender] >= balances[msg.sender] -
                 delayLockBeforeValues[msg.sender]
708            && delayLockTimes[msg.sender] > now
709            -> __return == balances[msg.sender] - delayLockBeforeValues[msg.sender]
710     */
```

Line 711-726 in File VHC.sol

```
711     function getMyUnlockValue() public view returns (uint256) {
712         uint256 myUnlockValue;
713         address addr = msg.sender;
714         if (delayLockTimes[addr] <= now) {
715             myUnlockValue = balances[addr].sub(delayLockValues[addr]);
716         } else {
717             myUnlockValue = balances[addr].sub(delayLockBeforeValues[addr]);
718         }
719
720         uint256 superUnlockValue = super.getMyUnlockValue();
721
722         if (myUnlockValue > superUnlockValue)
723             return superUnlockValue;
724         else
725             return myUnlockValue;
726     }
```

The code meets the specification

Formal Verification Request 33

If method completes, integer overflow would not happen.

28, May 2019
1315.28 ms

Line 761 in File VHC.sol

```
761     //@CTK NO_OVERFLOW
```

Line 777-779 in File VHC.sol

```
777     function transfer(address to, uint256 value) public returns (bool ret) {
778         return hintTransfer(to, value, "");
779     }
```

The code meets the specification

Formal Verification Request 34

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

28, May 2019
60.65 ms

Line 762 in File VHC.sol

```
762     //@CTK NO_BUF_OVERFLOW
```

Line 777-779 in File VHC.sol

```
777     function transfer(address to, uint256 value) public returns (bool ret) {
778         return hintTransfer(to, value, "");
779     }
```

The code meets the specification

Formal Verification Request 35

transfer correctness

28, May 2019
33937.85 ms

Line 764-776 in File VHC.sol

```
764     /*@CTK "transfer correctness"
765       @tag assume_completion
766       @post to != 0x0
767       @post to != address(this)
768       @post value <= balances[msg.sender]
769       @post (value <= balances[msg.sender] - delayLockValues[msg.sender])
770            || (value <= balances[msg.sender] - delayLockBeforeValues[msg.sender])
771       @post value <= balances[msg.sender] - lockValues[msg.sender]
772       @post (!locked || unlockAddrs[msg.sender])
```

```
773     @post to != msg.sender -> __post.balances[msg.sender] == balances[msg.sender] -
            value
774     @post to != msg.sender -> __post.balances[to] == balances[to] + value
775     @post to == msg.sender -> __post.balances[msg.sender] == balances[msg.sender]
776    */
```

Line 777-779 in File VHC.sol

```
777   function transfer(address to, uint256 value) public returns (bool ret) {
778       return hintTransfer(to, value, "");
779   }
```

The code meets the specification

Formal Verification Request 36

If method completes, integer overflow would not happen.

28, May 2019
1463.78 ms

Line 788 in File VHC.sol

```
788   //@CTK NO_OVERFLOW
```

Line 805-807 in File VHC.sol

```
805   function transferFrom(address from, address to, uint256 value) public returns (
          bool) {
806       return hintTransferFrom(from, to, value, "");
807   }
```

The code meets the specification

Formal Verification Request 37

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

28, May 2019
65.28 ms

Line 789 in File VHC.sol

```
789   //@CTK NO_BUF_OVERFLOW
```

Line 805-807 in File VHC.sol

```
805   function transferFrom(address from, address to, uint256 value) public returns (
          bool) {
806       return hintTransferFrom(from, to, value, "");
807   }
```

The code meets the specification

Formal Verification Request 38

transferFrom correctness

28, May 2019
48225.74 ms

Line 791-804 in File VHC.sol

```
791     /*@CTK "transferFrom correctness"
792       @tag assume_completion
793       @post to != 0x0
794       @post to != address(this)
795       @post value <= balances[from] && value <= allowed[from][msg.sender]
796       @post (value <= balances[from] - delayLockValues[from])
797            || (value <= balances[from] - delayLockBeforeValues[from])
798       @post value <= balances[from] - lockValues[from]
799       @post (!locked || unlockAddrs[from])
800       @post to != from -> __post.balances[from] == balances[from] - value
801       @post to != from -> __post.balances[to] == balances[to] + value
802       @post to == from -> __post.balances[from] == balances[from]
803       @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
804      */
```

Line 805-807 in File VHC.sol

```
805     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
806         return hintTransferFrom(from, to, value, "");
807     }
```

The code meets the specification

Formal Verification Request 39

approve correctness

28, May 2019
96.92 ms

Line 816-821 in File VHC.sol

```
816     /*@CTK "approve correctness"
817       @post !(__has_overflow)
818       @post !(__has_buf_overflow)
819       @post !(__has_assertion_failure)
820       @post __post.allowed[msg.sender][spender] == value
821      */
```

Line 822-824 in File VHC.sol

```
822     function approve(address spender, uint256 value) public returns (bool) {
823         return hintApprove(spender, value, "");
824     }
```

The code meets the specification

Formal Verification Request 40

increaseAllowance

28, May 2019
184.07 ms

Line 831-838 in File VHC.sol

```
831     /*@CTK increaseAllowance
832       @tag assume_completion
833       @post !(__has_overflow)
834       @post !(__has_buf_overflow)
835       @post !(__has_assertion_failure)
836       @post __post.allowed[msg.sender][spender] ==
837             allowed[msg.sender][spender] + addedValue
838      */
```

Line 839-841 in File VHC.sol

```
839     function increaseApproval(address spender, uint256 addedValue) public returns (
            bool) {
840         return hintIncreaseApproval(spender, addedValue, "");
841     }
```

The code meets the specification

Formal Verification Request 41

decreaseAllowance

28, May 2019
230.86 ms

Line 848-857 in File VHC.sol

```
848     /*@CTK decreaseAllowance
849       @tag assume_completion
850       @post !(__has_overflow)
851       @post !(__has_buf_overflow)
852       @post !(__has_assertion_failure)
853       @post allowed[msg.sender][spender] >= subtractedValue ->
854             __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] -
                  subtractedValue
855       @post allowed[msg.sender][spender] < subtractedValue ->
856             __post.allowed[msg.sender][spender] == 0
857      */
```

Line 858-860 in File VHC.sol

```
858     function decreaseApproval(address spender, uint256 subtractedValue) public returns
            (bool) {
859         return hintDecreaseApproval(spender, subtractedValue, "");
860     }
```

The code meets the specification

Formal Verification Request 42

hintMintTo

28, May 2019
379.11 ms

Line 880-890 in File VHC.sol

```
880     /*@CTK hintMintTo
881       @tag assume_completion
882       @post !(__has_overflow)
883       @post !(__has_buf_overflow)
884       @post !(__has_assertion_failure)
885       @post to != address(0)
886       @post owners[msg.sender] == true
887       @post __post.totalSupply_ == totalSupply_ + amount
888       @post __post.balances[to] == balances[to] + amount
889       @post ret == true
890      */
```

Line 891-894 in File VHC.sol

```
891     function hintMintTo(address to, uint256 amount, string note) onlyOwner public
            returns (bool ret) {
892         ret = mintTo(to, amount);
893         emit HINTMintTo(msg.sender, to, amount, note);
894     }
```

The code meets the specification

Formal Verification Request 43

hintBurnFrom

28, May 2019
438.72 ms

Line 909-920 in File VHC.sol

```
909     /*@CTK hintBurnFrom
910       @tag assume_completion
911       @pre balances[from] <= totalSupply_
912       @post !(__has_overflow)
913       @post !(__has_buf_overflow)
914       @post !(__has_assertion_failure)
915       @post owners[msg.sender] == true
916       @post value <= balances[from]
917       @post __post.totalSupply_ == totalSupply_ - value
918       @post __post.balances[from] == balances[from] - value
919       @post ret == true
920      */
```

Line 921-924 in File VHC.sol

```
921     function hintBurnFrom(address from, uint256 value, string note) onlyOwner public
            returns (bool ret) {
922         ret = burnFrom(from, value);
923         emit HINTBurnFrom(msg.sender, from, value, note);
```

```
924     }
```

The code meets the specification

Formal Verification Request 44

hintBurnWhenMoveToMainnet

28, May 2019
537.63 ms

Line 929-940 in File VHC.sol

```
929     /*@CTK hintBurnWhenMoveToMainnet
930       @tag assume_completion
931       @pre balances[burner] <= totalSupply_
932       @post !(__has_overflow)
933       @post !(__has_buf_overflow)
934       @post !(__has_assertion_failure)
935       @post owners[msg.sender] == true
936       @post value <= balances[burner]
937       @post __post.totalSupply_ == totalSupply_ - value
938       @post __post.balances[burner] == balances[burner] - value
939       @post ret == true
940      */
```

Line 941-944 in File VHC.sol

```
941     function hintBurnWhenMoveToMainnet(address burner, uint256 value, string note)
            onlyOwner public returns (bool ret) {
942         ret = hintBurnFrom(burner, value, note);
943         emit HINTBurnWhenMoveToMainnet(msg.sender, burner, value, note);
944     }
```

The code meets the specification

Formal Verification Request 45

If method completes, integer overflow would not happen.

28, May 2019
496.88 ms

Line 959 in File VHC.sol

```
959     //@CTK NO_OVERFLOW
```

Line 977-987 in File VHC.sol

```
977     function hintSell(
978         address from,
979         address to,
980         uint256 value,
981         string note
982     ) onlyOwner public returns (bool ret) {
983         require(to != address(this), "The receive address is the Contact Address of
                HINTToken. You cannot send money to this address.");
```

```
984
985         ret = hintTransferFrom(from, to, value, note);
986         emit HINTSell(from, msg.sender, to, value, note);
987     }
```

The code meets the specification

Formal Verification Request 46

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

28, May 2019
113.3 ms

Line 960 in File VHC.sol

```
960     //@CTK NO_BUF_OVERFLOW
```

Line 977-987 in File VHC.sol

```
977     function hintSell(
978         address from,
979         address to,
980         uint256 value,
981         string note
982     ) onlyOwner public returns (bool ret) {
983         require(to != address(this), "The receive address is the Contact Address of
                HINTToken. You cannot send money to this address.");
984
985         ret = hintTransferFrom(from, to, value, note);
986         emit HINTSell(from, msg.sender, to, value, note);
987     }
```

The code meets the specification

Formal Verification Request 47

hintSell correctness

28, May 2019
34046.72 ms

Line 962-976 in File VHC.sol

```
962     /*@CTK "hintSell correctness"
963       @tag assume_completion
964       @post to != 0x0
965       @post to != address(this)
966       @post owners[msg.sender] == true
967       @post value <= balances[from] && value <= allowed[from][msg.sender]
968       @post (value <= balances[from] - delayLockValues[from])
969            || (value <= balances[from] - delayLockBeforeValues[from])
970       @post value <= balances[from] - lockValues[from]
971       @post (!locked || unlockAddrs[from])
972       @post to != from -> __post.balances[from] == balances[from] - value
973       @post to != from -> __post.balances[to] == balances[to] + value
```

```
974      @post to == from -> __post.balances[from] == balances[from]
975      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
976     */
```

Line 977-987 in File VHC.sol

```
977    function hintSell(
978        address from,
979        address to,
980        uint256 value,
981        string note
982    ) onlyOwner public returns (bool ret) {
983        require(to != address(this), "The receive address is the Contact Address of
               HINTToken. You cannot send money to this address.");
984
985        ret = hintTransferFrom(from, to, value, note);
986        emit HINTSell(from, msg.sender, to, value, note);
987    }
```

The code meets the specification

Formal Verification Request 48

If method completes, integer overflow would not happen.

28, May 2019
492.5 ms

Line 1017 in File VHC.sol

```
1017    //@CTK NO_OVERFLOW
```

Line 1035-1045 in File VHC.sol

```
1035    function hintTransferToTeam(
1036        address from,
1037        address to,
1038        uint256 value,
1039        string note
1040    ) onlyOwner public returns (bool ret) {
1041        require(to != address(this), "The receive address is the Contact Address of
               HINTToken. You cannot send money to this address.");
1042
1043        ret = hintTransferFrom(from, to, value, note);
1044        emit HINTTransferToTeam(from, msg.sender, to, value, note);
1045    }
```

The code meets the specification

Formal Verification Request 49

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

28, May 2019
58.25 ms

Line 1018 in File VHC.sol

```
1018    //@CTK NO_BUF_OVERFLOW
```

Line 1035-1045 in File VHC.sol

```
1035    function hintTransferToTeam(
1036        address from,
1037        address to,
1038        uint256 value,
1039        string note
1040    ) onlyOwner public returns (bool ret) {
1041        require(to != address(this), "The receive address is the Contact Address of
            HINTToken. You cannot send money to this address.");
1042
1043        ret = hintTransferFrom(from, to, value, note);
1044        emit HINTTransferToTeam(from, msg.sender, to, value, note);
1045    }
```

The code meets the specification

Formal Verification Request 50

hintTransferToTeam correctness

28, May 2019
43971.24 ms

Line 1020-1034 in File VHC.sol

```
1020    /*@CTK "hintTransferToTeam correctness"
1021      @tag assume_completion
1022      @post to != 0x0
1023      @post to != address(this)
1024      @post owners[msg.sender] == true
1025      @post value <= balances[from] && value <= allowed[from][msg.sender]
1026      @post (value <= balances[from] - delayLockValues[from])
1027           || (value <= balances[from] - delayLockBeforeValues[from])
1028      @post value <= balances[from] - lockValues[from]
1029      @post (!locked || unlockAddrs[from])
1030      @post to != from -> __post.balances[from] == balances[from] - value
1031      @post to != from -> __post.balances[to] == balances[to] + value
1032      @post to == from -> __post.balances[from] == balances[from]
1033      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
1034     */
```

Line 1035-1045 in File VHC.sol

```
1035    function hintTransferToTeam(
1036        address from,
1037        address to,
1038        uint256 value,
1039        string note
1040    ) onlyOwner public returns (bool ret) {
1041        require(to != address(this), "The receive address is the Contact Address of
            HINTToken. You cannot send money to this address.");
1042
1043        ret = hintTransferFrom(from, to, value, note);
1044        emit HINTTransferToTeam(from, msg.sender, to, value, note);
1045    }
```

The code meets the specification

Formal Verification Request 51

If method completes, integer overflow would not happen.

28, May 2019
598.49 ms

Line 1050 in File VHC.sol

```
1050    //@CTK NO_OVERFLOW
```

Line 1068-1078 in File VHC.sol

```
1068    function hintTransferToPartner(
1069        address from,
1070        address to,
1071        uint256 value,
1072        string note
1073    ) onlyOwner public returns (bool ret) {
1074        require(to != address(this), "The receive address is the Contact Address of
                HINTToken. You cannot send money to this address.");
1075
1076        ret = hintTransferFrom(from, to, value, note);
1077        emit HINTTransferToPartner(from, msg.sender, to, value, note);
1078    }
```

The code meets the specification

Formal Verification Request 52

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

28, May 2019
54.28 ms

Line 1051 in File VHC.sol

```
1051    //@CTK NO_BUF_OVERFLOW
```

Line 1068-1078 in File VHC.sol

```
1068    function hintTransferToPartner(
1069        address from,
1070        address to,
1071        uint256 value,
1072        string note
1073    ) onlyOwner public returns (bool ret) {
1074        require(to != address(this), "The receive address is the Contact Address of
                HINTToken. You cannot send money to this address.");
1075
1076        ret = hintTransferFrom(from, to, value, note);
1077        emit HINTTransferToPartner(from, msg.sender, to, value, note);
1078    }
```

The code meets the specification

Formal Verification Request 53

hintTransferToPartner correctness

28, May 2019
40296.02 ms

Line 1053-1067 in File VHC.sol

```
1053    /*@CTK "hintTransferToPartner correctness"
1054      @tag assume_completion
1055      @post to != 0x0
1056      @post to != address(this)
1057      @post owners[msg.sender] == true
1058      @post value <= balances[from] && value <= allowed[from][msg.sender]
1059      @post (value <= balances[from] - delayLockValues[from])
1060           || (value <= balances[from] - delayLockBeforeValues[from])
1061      @post value <= balances[from] - lockValues[from]
1062      @post (!locked || unlockAddrs[from])
1063      @post to != from -> __post.balances[from] == balances[from] - value
1064      @post to != from -> __post.balances[to] == balances[to] + value
1065      @post to == from -> __post.balances[from] == balances[from]
1066      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
1067     */
```

Line 1068-1078 in File VHC.sol

```
1068    function hintTransferToPartner(
1069        address from,
1070        address to,
1071        uint256 value,
1072        string note
1073    ) onlyOwner public returns (bool ret) {
1074        require(to != address(this), "The receive address is the Contact Address of
            HINTToken. You cannot send money to this address.");
1075
1076        ret = hintTransferFrom(from, to, value, note);
1077        emit HINTTransferToPartner(from, msg.sender, to, value, note);
1078    }
```

The code meets the specification

Formal Verification Request 54

constructor correctness

28, May 2019
24.33 ms

Line 1150-1157 in File VHC.sol

```
1150    /*@CTK "constructor correctness"
1151      @tag assume_completion
1152      @post !(__has_overflow)
1153      @post !(__has_buf_overflow)
1154      @post !(__has_assertion_failure)
1155      @post __post.totalSupply_ == INITIAL_SUPPLY
1156      @post __post.balances[msg.sender] == __post.totalSupply_
```

```
1157     */
```

Line 1158-1162 in File VHC.sol

```
1158    constructor() public {
1159        totalSupply_ = INITIAL_SUPPLY;
1160        balances[msg.sender] = INITIAL_SUPPLY;
1161        emit Transfer(0x0, msg.sender, INITIAL_SUPPLY);
1162    }
```

The code meets the specification

Formal Verification Request 55

onHINTReceived correctness

28, May 2019
8.04 ms

Line 1211-1216 in File VHC.sol

```
1211    /*@CTK "onHINTReceived correctness"
1212      @post !(__has_overflow)
1213      @post !(__has_buf_overflow)
1214      @post !(__has_assertion_failure)
1215      @post __return == true
1216     */
```

Line 1217-1220 in File VHC.sol

```
1217    function onHINTReceived(address owner, address spender, uint256 value,
            HINTReceiveType receiveType) public returns (bool) {
1218        emit LogOnReceiveHINT("I receive HINT Token.", owner, spender, value,
                receiveType);
1219        return true;
1220    }
```

The code meets the specification

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File VHC.sol

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

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

TIMESTAMP_DEPENDENCY

Line 624 in File VHC.sol

```
624         if (delayLockTimes[msg.sender] <= now) {
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 649 in File VHC.sol

```
649             delayLockTimes[msg.sender] = now;
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 651 in File VHC.sol

```
651             require (delayLockTimes[msg.sender] <= now, "The remaining money in the
                    account cannot be unlocked continuously. You cannot renew until 12
                    hours after the first run.");
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 652 in File VHC.sol

```
652             delayLockTimes[msg.sender] = now + 12 hours;
```

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 714 in File VHC.sol

```
714         if (delayLockTimes[addr] <= now) {
```

"now" can be influenced by minors to some degree

Source Code with CertiK Labels

File VHC.sol

```
1  pragma solidity ^0.4.24;
2
3  /**
4   * @title ERC20Basic
5   * dev Simpler version of ERC20 interface
6   * See https://github.com/ethereum/EIPs/issues/179
7   */
8  contract ERC20Basic {
9      function totalSupply() public view returns (uint256);
10     function balanceOf(address who) public view returns (uint256);
11     function transfer(address to, uint256 value) public returns (bool);
12     event Transfer(address indexed from, address indexed to, uint256 value);
13 }
14
15 /**
16  * @title SafeMath
17  * dev Math operations with safety checks that throw on error
18  */
19 library SafeMath {
20
21     /**
22     * dev Multiplies two numbers, throws on overflow.
23     */
24     //@CTK FAIL NO_ASF
25     /*@CTK "SafeMath mul"
26       @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
27       @post !__reverted -> c == a * b
28       @post !__reverted == !__has_overflow
29       @post !(__has_buf_overflow)
30      */
31     function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
32         // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
33         // benefit is lost if 'b' is also tested.
34         // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
35         if (a == 0) {
36             return 0;
37         }
38
39         c = a * b;
40         assert(c / a == b);
41         return c;
42     }
43
44     /**
45     * dev Integer division of two numbers, truncating the quotient.
46     */
47     /*@CTK "SafeMath div"
48       @post b != 0 -> !__reverted
49       @post !__reverted -> __return == a / b
50       @post !__reverted -> !__has_overflow
51       @post !(__has_buf_overflow)
52       @post !__reverted -> !(__has_assertion_failure)
53     */
54     function div(uint256 a, uint256 b) internal pure returns (uint256) {
```

```
55          // assert(b > 0); // Solidity automatically throws when dividing by 0
56          // uint256 c = a / b;
57          // assert(a == b * c + a % b); // There is no case in which this doesn't hold
58          return a / b;
59      }
60
61      /**
62      * dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater
            than minuend).
63      */
64      //@CTK FAIL NO_ASF
65      /*@CTK "SafeMath sub"
66        @post (a < b) == __reverted
67        @post !__reverted -> __return == a - b
68        @post !__reverted -> !__has_overflow
69        @post !(__has_buf_overflow)
70      */
71      function sub(uint256 a, uint256 b) internal pure returns (uint256) {
72          assert(b <= a);
73          return a - b;
74      }
75
76      /**
77      * dev Adds two numbers, throws on overflow.
78      */
79      //@CTK FAIL NO_ASF
80      /*@CTK "SafeMath add"
81        @post (a + b < a || a + b < b) == __reverted
82        @post !__reverted -> c == a + b
83        @post !__reverted -> !__has_overflow
84        @post !(__has_buf_overflow)
85      */
86      function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
87          c = a + b;
88          assert(c >= a);
89          return c;
90      }
91  }
92
93  /**
94   * @title Basic token
95   * dev Basic version of StandardToken, with no allowances.
96   */
97  contract BasicToken is ERC20Basic {
98      using SafeMath for uint256;
99
100     mapping(address => uint256) balances;
101
102     uint256 totalSupply_;
103
104     /**
105     * dev Total number of tokens in existence
106     */
107     /*@CTK "totalSupply correctness"
108       @post __return == totalSupply_
109      */
110     function totalSupply() public view returns (uint256) {
111         return totalSupply_;
```

```
112     }
113
114     /**
115     * dev Transfer token for a specified address
116     * @param _to The address to transfer to.
117     * @param _value The amount to be transferred.
118     */
119     function transfer(address _to, uint256 _value) public returns (bool) {
120         require(_to != address(0), "Recipient address is zero address(0). Check the
                address again.");
121         require(_value <= balances[msg.sender], "The balance of account is insufficient
                .");
122
123         balances[msg.sender] = balances[msg.sender].sub(_value);
124         balances[_to] = balances[_to].add(_value);
125         emit Transfer(msg.sender, _to, _value);
126         return true;
127     }
128
129     /**
130     * dev Gets the balance of the specified address.
131     * @param _owner The address to query the the balance of.
132     * @return An uint256 representing the amount owned by the passed address.
133     */
134     /*@CTK "balanceOf correctness"
135       @post __return == balances[_owner]
136      */
137     function balanceOf(address _owner) public view returns (uint256) {
138         return balances[_owner];
139     }
140
141 }
142
143 /**
144  * @title ERC20 interface
145  * dev see https://github.com/ethereum/EIPs/issues/20
146  */
147 contract ERC20 is ERC20Basic {
148     function allowance(address owner, address spender)
149     public view returns (uint256);
150
151     function transferFrom(address from, address to, uint256 value)
152     public returns (bool);
153
154     function approve(address spender, uint256 value) public returns (bool);
155     event Approval(
156         address indexed owner,
157         address indexed spender,
158         uint256 value
159     );
160 }
161
162 /**
163  * @title Standard ERC20 token
164  *
165  * dev Implementation of the basic standard token.
166  * https://github.com/ethereum/EIPs/issues/20
```

```
167  * Based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master/
         smart_contract/FirstBloodToken.sol
168  */
169 contract StandardToken is ERC20, BasicToken {
170
171     mapping (address => mapping (address => uint256)) internal allowed;
172
173
174     /**
175      * dev Transfer tokens from one address to another
176      * @param _from address The address which you want to send tokens from
177      * @param _to address The address which you want to transfer to
178      * @param _value uint256 the amount of tokens to be transferred
179      */
180     function transferFrom(
181         address _from,
182         address _to,
183         uint256 _value
184     )
185     public
186     returns (bool)
187     {
188         require(_to != address(0), "Recipient address is zero address(0). Check the
                address again.");
189         require(_value <= balances[_from], "The balance of account is insufficient.");
190         require(_value <= allowed[_from][msg.sender], "Insufficient tokens approved
                from account owner.");
191
192         balances[_from] = balances[_from].sub(_value);
193         balances[_to] = balances[_to].add(_value);
194         allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
195         emit Transfer(_from, _to, _value);
196         return true;
197     }
198
199     /**
200      * dev Approve the passed address to spend the specified amount of tokens on
            behalf of msg.sender.
201      * Beware that changing an allowance with this method brings the risk that someone
             may use both the old
202      * and the new allowance by unfortunate transaction ordering. One possible
            solution to mitigate this
203      * race condition is to first reduce the spender's allowance to 0 and set the
            desired value afterwards:
204      * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
205      * @param _spender The address which will spend the funds.
206      * @param _value The amount of tokens to be spent.
207      */
208     function approve(address _spender, uint256 _value) public returns (bool) {
209         allowed[msg.sender][_spender] = _value;
210         emit Approval(msg.sender, _spender, _value);
211         return true;
212     }
213
214     /**
215      * dev Function to check the amount of tokens that an owner allowed to a spender.
216      * @param _owner address The address which owns the funds.
217      * @param _spender address The address which will spend the funds.
```

```
218      * @return A uint256 specifying the amount of tokens still available for the
             spender.
219      */
220     function allowance(
221         address _owner,
222         address _spender
223     )
224     public
225     view
226     returns (uint256)
227     {
228         return allowed[_owner][_spender];
229     }
230
231     /**
232      * dev Increase the amount of tokens that an owner allowed to a spender.
233      * approve should be called when allowed[_spender] == 0. To increment
234      * allowed value is better to use this function to avoid 2 calls (and wait until
235      * the first transaction is mined)
236      * From MonolithDAO Token.sol
237      * @param _spender The address which will spend the funds.
238      * @param _addedValue The amount of tokens to increase the allowance by.
239      */
240     function increaseApproval(
241         address _spender,
242         uint256 _addedValue
243     )
244     public
245     returns (bool)
246     {
247         allowed[msg.sender][_spender] = (
248         allowed[msg.sender][_spender].add(_addedValue));
249         emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
250         return true;
251     }
252
253     /**
254      * dev Decrease the amount of tokens that an owner allowed to a spender.
255      * approve should be called when allowed[_spender] == 0. To decrement
256      * allowed value is better to use this function to avoid 2 calls (and wait until
257      * the first transaction is mined)
258      * From MonolithDAO Token.sol
259      * @param _spender The address which will spend the funds.
260      * @param _subtractedValue The amount of tokens to decrease the allowance by.
261      */
262     function decreaseApproval(
263         address _spender,
264         uint256 _subtractedValue
265     )
266     public
267     returns (bool)
268     {
269         uint256 oldValue = allowed[msg.sender][_spender];
270         if (_subtractedValue > oldValue) {
271             allowed[msg.sender][_spender] = 0;
272         } else {
273             allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
274         }
```

```
275         emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
276         return true;
277     }
278 
279 }
280 
281 /**
282  * Utility library of inline functions on addresses
283  */
284 library AddressUtils {
285 
286     /**
287      * Returns whether the target address is a contract
288      * dev This function will return false if invoked during the constructor of a
            contract,
289      * as the code is not actually created until after the constructor finishes.
290      * @param addr address to check
291      * @return whether the target address is a contract
292      */
293     function isContract(address addr) internal view returns (bool) {
294         uint256 size;
295         // XXX Currently there is no better way to check if there is a contract in an
              address
296         // than to check the size of the code at that address.
297         // See https://ethereum.stackexchange.com/a/14016/36603
298         // for more details about how this works.
299         // TODO Check this again before the Serenity release, because all addresses
              will be
300         // contracts then.
301         // solium-disable-next-line security/no-inline-assembly
302         assembly { size := extcodesize(addr) }
303         return size > 0;
304     }
305 
306 }
307 
308 /**
309  * @title MultiOwnable
310  * dev
311  */
312 contract MultiOwnable {
313     using SafeMath for uint256;
314 
315     address public root; //                          superOwner          .
                                                .
316     address public superOwner;
317     mapping (address => bool) public owners;
318     address[] public ownerList;
319 
320     // for changeSuperOwnerByDAO
321     // mapping(address => mapping (address => bool)) public preSuperOwnerMap;
322     mapping(address => address) public candidateSuperOwnerMap;
323 
324 
325     event ChangedRoot(address newRoot);
326     event ChangedSuperOwner(address newSuperOwner);
327     event AddedNewOwner(address newOwner);
328     event DeletedOwner(address deletedOwner);
```

```
329
330     constructor() public {
331         root = msg.sender;
332         superOwner = msg.sender;
333         owners[root] = true;
334
335         ownerList.push(msg.sender);
336
337     }
338
339     modifier onlyRoot() {
340         require(msg.sender == root, "Root privilege is required.");
341         _;
342     }
343
344     modifier onlySuperOwner() {
345         require(msg.sender == superOwner, "SuperOwner priviledge is required.");
346         _;
347     }
348
349     modifier onlyOwner() {
350         require(owners[msg.sender], "Owner priviledge is required.");
351         _;
352     }
353
354     /**
355      * dev root      (root    root    superOwner
            .)
356      * dev                                              ,
                                              !
357      */
358     /*@CTK "changeRoot correctness"
359       @tag assume_completion
360       @post !(__has_overflow)
361       @post !(__has_buf_overflow)
362       @post !(__has_assertion_failure)
363       @post newRoot != 0x0
364       @post __post.root == newRoot
365      */
366     function changeRoot(address newRoot) onlyRoot public returns (bool) {
367         require(newRoot != address(0), "This address to be set is zero address(0).
              Check the input address.");
368
369         root = newRoot;
370
371         emit ChangedRoot(newRoot);
372         return true;
373     }
374
375     /**
376      * dev superOwner      (root    root    superOwner
                .)
377      * dev       superOwner                                    ,     superOwner
                                                  !
378      */
379     /*@CTK "changeSuperOwner correctness"
380       @tag assume_completion
381       @post !(__has_overflow)
```

```
382       @post !(__has_buf_overflow)
383       @post !(__has_assertion_failure)
384       @post msg.sender == root
385       @post newSuperOwner != 0x0
386       @post __post.superOwner == newSuperOwner
387     */
388     function changeSuperOwner(address newSuperOwner) onlyRoot public returns (bool) {
389         require(newSuperOwner != address(0), "This address to be set is zero address(0)
                . Check the input address.");
390 
391         superOwner = newSuperOwner;
392 
393         emit ChangedSuperOwner(newSuperOwner);
394         return true;
395     }
396 
397     /**
398      * dev owner      1/2                    superOwner                    .
399      */
400     function changeSuperOwnerByDAO(address newSuperOwner) onlyOwner public returns (
            bool) {
401         require(newSuperOwner != address(0), "This address to be set is zero address(0)
                . Check the input address.");
402         require(newSuperOwner != candidateSuperOwnerMap[msg.sender], "You have already
                voted for this account.");
403 
404         candidateSuperOwnerMap[msg.sender] = newSuperOwner;
405 
406         uint8 votingNumForSuperOwner = 0;
407         uint8 i = 0;
408 
409         for (i = 0; i < ownerList.length; i++) {
410             if (candidateSuperOwnerMap[ownerList[i]] == newSuperOwner)
411                 votingNumForSuperOwner++;
412         }
413 
414         if (votingNumForSuperOwner > ownerList.length / 2) { //                DAO
                    => superOwner
415             superOwner = newSuperOwner;
416 
417             //
418             for (i = 0; i < ownerList.length; i++) {
419                 delete candidateSuperOwnerMap[ownerList[i]];
420             }
421 
422             emit ChangedSuperOwner(newSuperOwner);
423         }
424 
425         return true;
426     }
427 
428     //@CTK NO_BUF_OVERFLOW
429     //@CTK NO_ASF
430     /*@CTK "newOwner correctness"
431       @tag assume_completion
432       @post msg.sender == superOwner
433       @post owner != 0x0
434       @post owners[owner] == false
```

```
435       @post __post.ownerList.length == ownerList.length + 1
436       @post __post.ownerList[ownerList.length] == owner
437       @post __post.owners[owner] == true
438      */
439     function newOwner(address owner) onlySuperOwner public returns (bool) {
440         require(owner != address(0), "This address to be set is zero address(0). Check
                the input address.");
441         require(!owners[owner], "This address is already registered.");
442 
443         owners[owner] = true;
444         ownerList.push(owner);
445 
446         emit AddedNewOwner(owner);
447         return true;
448     }
449 
450     function deleteOwner(address owner) onlySuperOwner public returns (bool) {
451         require(owners[owner], "This input address is not a super owner.");
452         delete owners[owner];
453 
454         for (uint256 i = 0; i < ownerList.length; i++) {
455             if (ownerList[i] == owner) {
456                 ownerList[i] = ownerList[ownerList.length.sub(1)];
457                 ownerList.length = ownerList.length.sub(1);
458                 break;
459             }
460         }
461 
462         emit DeletedOwner(owner);
463         return true;
464     }
465 }
466 
467 /**
468  * @title Lockable token
469  */
470 contract LockableToken is StandardToken, MultiOwnable {
471     bool public locked = true;
472     uint256 public constant LOCK_MAX = uint256(-1);
473 
474     /**
475      * dev
476      */
477     mapping(address => bool) public unlockAddrs;
478 
479     /**
480      * dev              lock value
481      * dev -       0        :          0                                    .
482      * dev -       LOCK_MAX        :          uint256
                .
483      */
484     mapping(address => uint256) public lockValues;
485 
486     event Locked(bool locked, string note);
487     event LockedTo(address indexed addr, bool locked, string note);
488     event SetLockValue(address indexed addr, uint256 value, string note);
489 
490     constructor() public {
```

```
491         unlockTo(msg.sender, "");
492     }
493
494     modifier checkUnlock (address addr, uint256 value) {
495         require(!locked || unlockAddrs[addr], "The account is currently locked.");
496         require(balances[addr] >= value, "Transferable limit exceeded. Check the status
                of the lock value.");
497         require(balances[addr] - value >= lockValues[addr], "Transferable limit
                exceeded. Check the status of the lock value.");
498         _;
499     }
500
501     /*@CTK "lock correctness"
502       @tag assume_completion
503       @post !(__has_overflow)
504       @post !(__has_buf_overflow)
505       @post !(__has_assertion_failure)
506       @post owners[msg.sender] == true
507       @post __post.locked == true
508      */
509     function lock(string note) onlyOwner public {
510         locked = true;
511         emit Locked(locked, note);
512     }
513
514     /*@CTK "unlock correctness"
515       @tag assume_completion
516       @post !(__has_overflow)
517       @post !(__has_buf_overflow)
518       @post !(__has_assertion_failure)
519       @post owners[msg.sender] == true
520       @post __post.locked == false
521      */
522     function unlock(string note) onlyOwner public {
523         locked = false;
524         emit Locked(locked, note);
525     }
526
527     /*@CTK "lockTo correctness"
528       @tag assume_completion
529       @pre LOCK_MAX == 255
530       @post !(__has_overflow)
531       @post !(__has_buf_overflow)
532       @post !(__has_assertion_failure)
533       @post owners[msg.sender] == true
534       @post __post.lockValues[addr] == 255
535       @post __post.unlockAddrs[addr] == false
536      */
537     function lockTo(address addr, string note) onlyOwner public {
538         setLockValue(addr, LOCK_MAX, note);
539         unlockAddrs[addr] = false;
540
541         emit LockedTo(addr, true, note);
542     }
543
544     /*@CTK "unlockTo correctness"
545       @tag assume_completion
546       @pre LOCK_MAX == 255
```

```
547       @post !(__has_overflow)
548       @post !(__has_buf_overflow)
549       @post !(__has_assertion_failure)
550       @post owners[msg.sender] == true
551       @post lockValues[addr] == 255 -> __post.lockValues[addr] == 0
552       @post __post.unlockAddrs[addr] == true
553     */
554     function unlockTo(address addr, string note) onlyOwner public {
555         if (lockValues[addr] == LOCK_MAX)
556             setLockValue(addr, 0, note);
557         unlockAddrs[addr] = true;
558 
559         emit LockedTo(addr, false, note);
560     }
561 
562     /*@CTK "setLockValue correctness"
563       @tag assume_completion
564       @post !(__has_overflow)
565       @post !(__has_buf_overflow)
566       @post !(__has_assertion_failure)
567       @post owners[msg.sender] == true
568       @post __post.lockValues[addr] == value
569     */
570     function setLockValue(address addr, uint256 value, string note) onlyOwner public {
571         lockValues[addr] = value;
572         emit SetLockValue(addr, value, note);
573     }
574 
575     /**
576      * dev                                  .
577      */
578     /*@CTK "getMyUnlockValue correctness"
579       @tag assume_completion
580       @post !(__has_overflow)
581       @post !(__has_buf_overflow)
582       @post !(__has_assertion_failure)
583       @post locked && !unlockAddrs[msg.sender]
584             -> __return == 0
585       @post !locked && (balances[msg.sender] > lockValues[msg.sender])
586             -> __return == balances[msg.sender] - lockValues[msg.sender]
587       @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] > lockValues[
              msg.sender])
588             -> __return == balances[msg.sender] - lockValues[msg.sender]
589       @post !locked && (balances[msg.sender] <= lockValues[msg.sender])
590             -> __return == 0
591       @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] <= lockValues[
              msg.sender])
592             -> __return == 0
593     */
594     function getMyUnlockValue() public view returns (uint256) {
595         address addr = msg.sender;
596         if ((!locked || unlockAddrs[addr]) && balances[addr] > lockValues[addr])
597             return balances[addr].sub(lockValues[addr]);
598         else
599             return 0;
600     }
601 
```

```
602     function transfer(address to, uint256 value) checkUnlock(msg.sender, value) public
            returns (bool) {
603         return super.transfer(to, value);
604     }
605
606     function transferFrom(address from, address to, uint256 value) checkUnlock(from,
            value) public returns (bool) {
607         return super.transferFrom(from, to, value);
608     }
609 }
610
611 /**
612  * @title DelayLockableToken
613  * dev                                          lock                .
                                          12                          .
614  */
615 contract DelayLockableToken is LockableToken {
616     mapping(address => uint256) public delayLockValues;
617     mapping(address => uint256) public delayLockBeforeValues;
618     mapping(address => uint256) public delayLockTimes;
619
620     event SetDelayLockValue(address indexed addr, uint256 value, uint256 time);
621
622     modifier checkDelayUnlock (address addr, uint256 value) {
623         require(balances[addr] >= value);
624         if (delayLockTimes[msg.sender] <= now) {
625             require (balances[addr] - value >= delayLockValues[addr], "Transferable
                    limit exceeded. Change the balance lock value first and then use it");
626         } else {
627             require (balances[addr] - value >= delayLockBeforeValues[addr], "
                    Transferable limit exceeded. Please note that the residual lock value
                    has changed and it will take 12 hours to apply.");
628         }
629         _;
630     }
631
632     /**
633      * dev                                    .                                    ,
                            12                          .
634      */
635     //@CTK FAIL NO_OVERFLOW
636     //@CTK NO_BUF_OVERFLOW
637     //@CTK NO_ASF
638     /*@CTK "delayLock correctness"
639       @tag assume_completion
640       @post value <= balances[msg.sender]
641       @post value < delayLockValues[msg.sender]
642            -> __post.delayLockBeforeValues[msg.sender] == delayLockValues[msg.sender]
643       @post __post.delayLockValues[msg.sender] == value
644      */
645     function delayLock(uint256 value) public returns (bool) {
646         require (value <= balances[msg.sender], "Your balance is insufficient.");
647
648         if (value >= delayLockValues[msg.sender])
649             delayLockTimes[msg.sender] = now;
650         else {
651             require (delayLockTimes[msg.sender] <= now, "The remaining money in the
                    account cannot be unlocked continuously. You cannot renew until 12
```

```
                hours after the first run.");
652             delayLockTimes[msg.sender] = now + 12 hours;
653             delayLockBeforeValues[msg.sender] = delayLockValues[msg.sender];
654         }
655
656         delayLockValues[msg.sender] = value;
657
658         emit SetDelayLockValue(msg.sender, value, delayLockTimes[msg.sender]);
659         return true;
660     }
661
662     /**
663      * dev                                          .
664      */
665     //@CTK FAIL NO_OVERFLOW
666     //@CTK NO_BUF_OVERFLOW
667     //@CTK NO_ASF
668     /*@CTK "delayUnlock correctness"
669       @tag assume_completion
670       @post delayLockValues[msg.sender] != 0
671             -> __post.delayLockBeforeValues[msg.sender] == delayLockValues[msg.sender]
672       @post __post.delayLockValues[msg.sender] == 0
673      */
674     function delayUnlock() public returns (bool) {
675         return delayLock(0);
676     }
677
678     /**
679      * dev                                      .
680      */
681     //@CTK NO_OVERFLOW
682     //@CTK NO_BUF_OVERFLOW
683     //@CTK FAIL NO_ASF
684     /*@CTK "DelayLockableToken getMyUnlockValue correctness"
685       @tag assume_completion
686       @pre balances[msg.sender] >= delayLockValues[msg.sender]
687       @pre balances[msg.sender] >= delayLockBeforeValues[msg.sender]
688       @post locked && !unlockAddrs[msg.sender]
689             -> __return == 0
690       @post !locked && (balances[msg.sender] <= lockValues[msg.sender])
691             -> __return == 0
692       @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] <= lockValues[
              msg.sender])
693             -> __return == 0
694       @post !locked && (balances[msg.sender] > lockValues[msg.sender])
695             && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                  delayLockValues[msg.sender]
696             && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                  delayLockBeforeValues[msg.sender]
697             -> __return == balances[msg.sender] - lockValues[msg.sender]
698       @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] > lockValues[
              msg.sender])
699             && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                  delayLockValues[msg.sender]
700             && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                  delayLockBeforeValues[msg.sender]
701             -> __return == balances[msg.sender] - lockValues[msg.sender]
702       @post !locked && (balances[msg.sender] > lockValues[msg.sender])
```

```
703             && balances[msg.sender] - lockValues[msg.sender] >= balances[msg.sender] -
                    delayLockValues[msg.sender]
704             && delayLockTimes[msg.sender] <= now
705             -> __return == balances[msg.sender] - delayLockValues[msg.sender]
706         @post !locked && (balances[msg.sender] > lockValues[msg.sender])
707             && balances[msg.sender] - lockValues[msg.sender] >= balances[msg.sender] -
                    delayLockBeforeValues[msg.sender]
708             && delayLockTimes[msg.sender] > now
709             -> __return == balances[msg.sender] - delayLockBeforeValues[msg.sender]
710     */
711     function getMyUnlockValue() public view returns (uint256) {
712         uint256 myUnlockValue;
713         address addr = msg.sender;
714         if (delayLockTimes[addr] <= now) {
715             myUnlockValue = balances[addr].sub(delayLockValues[addr]);
716         } else {
717             myUnlockValue = balances[addr].sub(delayLockBeforeValues[addr]);
718         }
719 
720         uint256 superUnlockValue = super.getMyUnlockValue();
721 
722         if (myUnlockValue > superUnlockValue)
723             return superUnlockValue;
724         else
725             return myUnlockValue;
726     }
727 
728     function transfer(address to, uint256 value) checkDelayUnlock(msg.sender, value)
            public returns (bool) {
729         return super.transfer(to, value);
730     }
731 
732     function transferFrom(address from, address to, uint256 value) checkDelayUnlock(
            from, value) public returns (bool) {
733         return super.transferFrom(from, to, value);
734     }
735 }
736 
737 /**
738  * @title HINTBaseToken
739  * dev                                                    .
740  */
741 contract HINTBaseToken is DelayLockableToken {
742     event HINTTransfer(address indexed from, address indexed to, uint256 value, string
             note);
743     event HINTTransferFrom(address indexed owner, address indexed spender, address
            indexed to, uint256 value, string note);
744     event HINTApproval(address indexed owner, address indexed spender, uint256 value,
            string note);
745 
746     event HINTMintTo(address indexed controller, address indexed to, uint256 amount,
            string note);
747     event HINTBurnFrom(address indexed controller, address indexed from, uint256 value
            , string note);
748 
749     event HINTBurnWhenMoveToMainnet(address indexed controller, address indexed from,
            uint256 value, string note);
750 
```

```
751     event HINTSell(address indexed owner, address indexed spender, address indexed to,
            uint256 value, string note);
752     event HINTSellByOtherCoin(address indexed owner, address indexed spender, address
            indexed to, uint256 value, uint256 processIdHash, uint256 userIdHash, string
            note);
753
754     event HINTTransferToTeam(address indexed owner, address indexed spender, address
            indexed to, uint256 value, string note);
755     event HINTTransferToPartner(address indexed owner, address indexed spender,
            address indexed to, uint256 value, string note);
756
757     event HINTTransferToEcosystem(address indexed owner, address indexed spender,
            address indexed to, uint256 value, uint256 processIdHash, uint256 userIdHash,
            string note);
758     event HINTTransferToBounty(address indexed owner, address indexed spender, address
            indexed to, uint256 value, uint256 processIdHash, uint256 userIdHash, string
            note);
759
760     // ERC20                                super                                hint~
                                    .
761     //@CTK NO_OVERFLOW
762     //@CTK NO_BUF_OVERFLOW
763     //CTK FAIL NO_ASF
764     /*@CTK "transfer correctness"
765       @tag assume_completion
766       @post to != 0x0
767       @post to != address(this)
768       @post value <= balances[msg.sender]
769       @post (value <= balances[msg.sender] - delayLockValues[msg.sender])
770             || (value <= balances[msg.sender] - delayLockBeforeValues[msg.sender])
771       @post value <= balances[msg.sender] - lockValues[msg.sender]
772       @post (!locked || unlockAddrs[msg.sender])
773       @post to != msg.sender -> __post.balances[msg.sender] == balances[msg.sender] -
              value
774       @post to != msg.sender -> __post.balances[to] == balances[to] + value
775       @post to == msg.sender -> __post.balances[msg.sender] == balances[msg.sender]
776      */
777     function transfer(address to, uint256 value) public returns (bool ret) {
778         return hintTransfer(to, value, "");
779     }
780
781     function hintTransfer(address to, uint256 value, string note) public returns (bool
            ret) {
782         require(to != address(this), "The receive address is the Contact Address of
                HINTToken. You cannot send money to this address.");
783
784         ret = super.transfer(to, value);
785         emit HINTTransfer(msg.sender, to, value, note);
786     }
787
788     //@CTK NO_OVERFLOW
789     //@CTK NO_BUF_OVERFLOW
790     //CTK FAIL NO_ASF
791     /*@CTK "transferFrom correctness"
792       @tag assume_completion
793       @post to != 0x0
794       @post to != address(this)
795       @post value <= balances[from] && value <= allowed[from][msg.sender]
```

```
796       @post (value <= balances[from] - delayLockValues[from])
797            || (value <= balances[from] - delayLockBeforeValues[from])
798       @post value <= balances[from] - lockValues[from]
799       @post (!locked || unlockAddrs[from])
800       @post to != from -> __post.balances[from] == balances[from] - value
801       @post to != from -> __post.balances[to] == balances[to] + value
802       @post to == from -> __post.balances[from] == balances[from]
803       @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
804     */
805     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
806         return hintTransferFrom(from, to, value, "");
807     }
808
809     function hintTransferFrom(address from, address to, uint256 value, string note)
            public returns (bool ret) {
810         require(to != address(this), "The receive address is the Contact Address of
                HINTToken. You cannot send money to this address.");
811
812         ret = super.transferFrom(from, to, value);
813         emit HINTTransferFrom(from, msg.sender, to, value, note);
814     }
815
816     /*@CTK "approve correctness"
817       @post !(__has_overflow)
818       @post !(__has_buf_overflow)
819       @post !(__has_assertion_failure)
820       @post __post.allowed[msg.sender][spender] == value
821     */
822     function approve(address spender, uint256 value) public returns (bool) {
823         return hintApprove(spender, value, "");
824     }
825
826     function hintApprove(address spender, uint256 value, string note) public returns (
            bool ret) {
827         ret = super.approve(spender, value);
828         emit HINTApproval(msg.sender, spender, value, note);
829     }
830
831     /*@CTK increaseAllowance
832       @tag assume_completion
833       @post !(__has_overflow)
834       @post !(__has_buf_overflow)
835       @post !(__has_assertion_failure)
836       @post __post.allowed[msg.sender][spender] ==
837            allowed[msg.sender][spender] + addedValue
838     */
839     function increaseApproval(address spender, uint256 addedValue) public returns (
            bool) {
840         return hintIncreaseApproval(spender, addedValue, "");
841     }
842
843     function hintIncreaseApproval(address spender, uint256 addedValue, string note)
            public returns (bool ret) {
844         ret = super.increaseApproval(spender, addedValue);
845         emit HINTApproval(msg.sender, spender, allowed[msg.sender][spender], note);
846     }
847
```

```
848     /*@CTK decreaseAllowance
849       @tag assume_completion
850       @post !(__has_overflow)
851       @post !(__has_buf_overflow)
852       @post !(__has_assertion_failure)
853       @post allowed[msg.sender][spender] >= subtractedValue ->
854             __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] -
                  subtractedValue
855       @post allowed[msg.sender][spender] < subtractedValue ->
856             __post.allowed[msg.sender][spender] == 0
857      */
858     function decreaseApproval(address spender, uint256 subtractedValue) public returns
           (bool) {
859         return hintDecreaseApproval(spender, subtractedValue, "");
860     }
861
862     function hintDecreaseApproval(address spender, uint256 subtractedValue, string
          note) public returns (bool ret) {
863         ret = super.decreaseApproval(spender, subtractedValue);
864         emit HINTApproval(msg.sender, spender, allowed[msg.sender][spender], note);
865     }
866
867     /**
868      * dev                    .                                    .
869      */
870     function mintTo(address to, uint256 amount) internal returns (bool) {
871         require(to != address(0x0), "This address to be set is zero address(0). Check
                the input address.");
872
873         totalSupply_ = totalSupply_.add(amount);
874         balances[to] = balances[to].add(amount);
875
876         emit Transfer(address(0), to, amount);
877         return true;
878     }
879
880     /*@CTK hintMintTo
881       @tag assume_completion
882       @post !(__has_overflow)
883       @post !(__has_buf_overflow)
884       @post !(__has_assertion_failure)
885       @post to != address(0)
886       @post owners[msg.sender] == true
887       @post __post.totalSupply_ == totalSupply_ + amount
888       @post __post.balances[to] == balances[to] + amount
889       @post ret == true
890      */
891     function hintMintTo(address to, uint256 amount, string note) onlyOwner public
           returns (bool ret) {
892         ret = mintTo(to, amount);
893         emit HINTMintTo(msg.sender, to, amount, note);
894     }
895
896     /**
897      * dev             .                                    .
898      */
899     function burnFrom(address from, uint256 value) internal returns (bool) {
900         require(value <= balances[from], "Your balance is insufficient.");
```

```
901
902         balances[from] = balances[from].sub(value);
903         totalSupply_ = totalSupply_.sub(value);
904
905         emit Transfer(from, address(0), value);
906         return true;
907     }
908
909     /*@CTK hintBurnFrom
910       @tag assume_completion
911       @pre balances[from] <= totalSupply_
912       @post !(__has_overflow)
913       @post !(__has_buf_overflow)
914       @post !(__has_assertion_failure)
915       @post owners[msg.sender] == true
916       @post value <= balances[from]
917       @post __post.totalSupply_ == totalSupply_ - value
918       @post __post.balances[from] == balances[from] - value
919       @post ret == true
920      */
921     function hintBurnFrom(address from, uint256 value, string note) onlyOwner public
          returns (bool ret) {
922         ret = burnFrom(from, value);
923         emit HINTBurnFrom(msg.sender, from, value, note);
924     }
925
926     /**
927      * dev                                          .
928      */
929     /*@CTK hintBurnWhenMoveToMainnet
930       @tag assume_completion
931       @pre balances[burner] <= totalSupply_
932       @post !(__has_overflow)
933       @post !(__has_buf_overflow)
934       @post !(__has_assertion_failure)
935       @post owners[msg.sender] == true
936       @post value <= balances[burner]
937       @post __post.totalSupply_ == totalSupply_ - value
938       @post __post.balances[burner] == balances[burner] - value
939       @post ret == true
940      */
941     function hintBurnWhenMoveToMainnet(address burner, uint256 value, string note)
          onlyOwner public returns (bool ret) {
942         ret = hintBurnFrom(burner, value, note);
943         emit HINTBurnWhenMoveToMainnet(msg.sender, burner, value, note);
944     }
945
946     function hintBatchBurnWhenMoveToMainnet(address[] burners, uint256[] values,
          string note) onlyOwner public returns (bool ret) {
947         uint256 length = burners.length;
948         require(length == values.length, "The size of \'burners\' and \'values\' array
              is different.");
949
950         ret = true;
951         for (uint256 i = 0; i < length; i++) {
952             ret = ret && hintBurnWhenMoveToMainnet(burners[i], values[i], note);
953         }
954     }
```

```
955
956     /**
957      * dev          HINT
958      */
959     //@CTK NO_OVERFLOW
960     //@CTK NO_BUF_OVERFLOW
961     //CTK FAIL NO_ASF
962     /*@CTK "hintSell correctness"
963       @tag assume_completion
964       @post to != 0x0
965       @post to != address(this)
966       @post owners[msg.sender] == true
967       @post value <= balances[from] && value <= allowed[from][msg.sender]
968       @post (value <= balances[from] - delayLockValues[from])
969            || (value <= balances[from] - delayLockBeforeValues[from])
970       @post value <= balances[from] - lockValues[from]
971       @post (!locked || unlockAddrs[from])
972       @post to != from -> __post.balances[from] == balances[from] - value
973       @post to != from -> __post.balances[to] == balances[to] + value
974       @post to == from -> __post.balances[from] == balances[from]
975       @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
976      */
977     function hintSell(
978         address from,
979         address to,
980         uint256 value,
981         string note
982     ) onlyOwner public returns (bool ret) {
983         require(to != address(this), "The receive address is the Contact Address of
                HINTToken. You cannot send money to this address.");
984
985         ret = hintTransferFrom(from, to, value, note);
986         emit HINTSell(from, msg.sender, to, value, note);
987     }
988
989     /**
990      * dev                                    HINT
991      * dev EOA
                            . (                        )
992      */
993     function hintBatchSellByOtherCoin(
994         address from,
995         address[] to,
996         uint256[] values,
997         uint256 processIdHash,
998         uint256[] userIdHash,
999         string note
1000    ) onlyOwner public returns (bool ret) {
1001        uint256 length = to.length;
1002        require(length == values.length, "The size of \'to\' and \'values\' array is
                different.");
1003        require(length == userIdHash.length, "The size of \'to\' and \'userIdHash\'
                array is different.");
1004
1005        ret = true;
1006        for (uint256 i = 0; i < length; i++) {
1007            require(to[i] != address(this), "The receive address is the Contact Address
                    of HINTToken. You cannot send money to this address.");
```

```
1008
1009            ret = ret && hintTransferFrom(from, to[i], values[i], note);
1010            emit HINTSellByOtherCoin(from, msg.sender, to[i], values[i], processIdHash,
                    userIdHash[i], note);
1011        }
1012    }
1013
1014    /**
1015     * dev
1016     */
1017    //@CTK NO_OVERFLOW
1018    //@CTK NO_BUF_OVERFLOW
1019    //CTK NO_ASF
1020    /*@CTK "hintTransferToTeam correctness"
1021      @tag assume_completion
1022      @post to != 0x0
1023      @post to != address(this)
1024      @post owners[msg.sender] == true
1025      @post value <= balances[from] && value <= allowed[from][msg.sender]
1026      @post (value <= balances[from] - delayLockValues[from])
1027           || (value <= balances[from] - delayLockBeforeValues[from])
1028      @post value <= balances[from] - lockValues[from]
1029      @post (!locked || unlockAddrs[from])
1030      @post to != from -> __post.balances[from] == balances[from] - value
1031      @post to != from -> __post.balances[to] == balances[to] + value
1032      @post to == from -> __post.balances[from] == balances[from]
1033      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
1034     */
1035    function hintTransferToTeam(
1036        address from,
1037        address to,
1038        uint256 value,
1039        string note
1040    ) onlyOwner public returns (bool ret) {
1041        require(to != address(this), "The receive address is the Contact Address of
                HINTToken. You cannot send money to this address.");
1042
1043        ret = hintTransferFrom(from, to, value, note);
1044        emit HINTTransferToTeam(from, msg.sender, to, value, note);
1045    }
1046
1047    /**
1048     * dev
1049     */
1050    //@CTK NO_OVERFLOW
1051    //@CTK NO_BUF_OVERFLOW
1052    //CTK FAIL NO_ASF
1053    /*@CTK "hintTransferToPartner correctness"
1054      @tag assume_completion
1055      @post to != 0x0
1056      @post to != address(this)
1057      @post owners[msg.sender] == true
1058      @post value <= balances[from] && value <= allowed[from][msg.sender]
1059      @post (value <= balances[from] - delayLockValues[from])
1060           || (value <= balances[from] - delayLockBeforeValues[from])
1061      @post value <= balances[from] - lockValues[from]
1062      @post (!locked || unlockAddrs[from])
1063      @post to != from -> __post.balances[from] == balances[from] - value
```

```
1064      @post to != from -> __post.balances[to] == balances[to] + value
1065      @post to == from -> __post.balances[from] == balances[from]
1066      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
1067     */
1068    function hintTransferToPartner(
1069        address from,
1070        address to,
1071        uint256 value,
1072        string note
1073    ) onlyOwner public returns (bool ret) {
1074        require(to != address(this), "The receive address is the Contact Address of
              HINTToken. You cannot send money to this address.");
1075
1076        ret = hintTransferFrom(from, to, value, note);
1077        emit HINTTransferToPartner(from, msg.sender, to, value, note);
1078    }
1079
1080    /**
1081     * dev              (                                    )      HINT
1082     * dev EOA
                            . (                          )
1083     */
1084    function hintBatchTransferToEcosystem(
1085        address from, address[] to,
1086        uint256[] values,
1087        uint256 processIdHash,
1088        uint256[] userIdHash,
1089        string note
1090    ) onlyOwner public returns (bool ret) {
1091        uint256 length = to.length;
1092        require(length == values.length, "The size of \'to\' and \'values\' array is
              different.");
1093        require(length == userIdHash.length, "The size of \'to\' and \'userIdHash\'
              array is different.");
1094
1095        ret = true;
1096        for (uint256 i = 0; i < length; i++) {
1097            require(to[i] != address(this), "The receive address is the Contact Address
                  of HINTToken. You cannot send money to this address.");
1098
1099            ret = ret && hintTransferFrom(from, to[i], values[i], note);
1100            emit HINTTransferToEcosystem(from, msg.sender, to[i], values[i],
                  processIdHash, userIdHash[i], note);
1101        }
1102    }
1103
1104    /**
1105     * dev                          HINT
1106     * dev EOA
                            . (                          )
1107     */
1108    function hintBatchTransferToBounty(
1109        address from,
1110        address[] to,
1111        uint256[] values,
1112        uint256 processIdHash,
1113        uint256[] userIdHash,
1114        string note
```

```
1115    ) onlyOwner public returns (bool ret) {
1116        uint256 length = to.length;
1117        require(to.length == values.length, "The size of \'to\' and \'values\' array is
                different.");
1118
1119        ret = true;
1120        for (uint256 i = 0; i < length; i++) {
1121            require(to[i] != address(this), "The receive address is the Contact Address
                    of HINTToken. You cannot send money to this address.");
1122
1123            ret = ret && hintTransferFrom(from, to[i], values[i], note);
1124            emit HINTTransferToBounty(from, msg.sender, to[i], values[i], processIdHash
                    , userIdHash[i], note);
1125        }
1126    }
1127
1128    function destroy() onlyRoot public {
1129        selfdestruct(root);
1130    }
1131 }
1132
1133 /**
1134  * @title HINTToken
1135  */
1136 contract HINTToken is HINTBaseToken {
1137    using AddressUtils for address;
1138
1139    event TransferedToHINTDapp(
1140        address indexed owner,
1141        address indexed spender,
1142        address indexed to, uint256 value, HINTReceiver.HINTReceiveType receiveType);
1143
1144    string public constant name = "HINT Token";
1145    string public constant symbol = "HINT";
1146    uint8 public constant decimals = 18;
1147
1148    uint256 public constant INITIAL_SUPPLY = 1e9 * (10 ** uint256(decimals));
1149
1150    /*@CTK "constructor correctness"
1151      @tag assume_completion
1152      @post !(__has_overflow)
1153      @post !(__has_buf_overflow)
1154      @post !(__has_assertion_failure)
1155      @post __post.totalSupply_ == INITIAL_SUPPLY
1156      @post __post.balances[msg.sender] == __post.totalSupply_
1157     */
1158    constructor() public {
1159        totalSupply_ = INITIAL_SUPPLY;
1160        balances[msg.sender] = INITIAL_SUPPLY;
1161        emit Transfer(0x0, msg.sender, INITIAL_SUPPLY);
1162    }
1163
1164    function hintTransfer(address to, uint256 value, string note) public returns (bool
             ret) {
1165        ret = super.hintTransfer(to, value, note);
1166        postTransfer(msg.sender, msg.sender, to, value, HINTReceiver.HINTReceiveType.
                HINT_TRANSFER);
1167    }
```

```
1168
1169        function hintTransferFrom(address from, address to, uint256 value, string note)
                public returns (bool ret) {
1170            ret = super.hintTransferFrom(from, to, value, note);
1171            postTransfer(from, msg.sender, to, value, HINTReceiver.HINTReceiveType.
                    HINT_TRANSFER);
1172        }
1173
1174        function postTransfer(address owner, address spender, address to, uint256 value,
                HINTReceiver.HINTReceiveType receiveType) internal returns (bool) {
1175            if (to.isContract()) {
1176                bool callOk = address(to).call(bytes4(keccak256("onHINTReceived(address,
                        address,uint256,uint8)")), owner, spender, value, receiveType);
1177                if (callOk) {
1178                    emit TransferedToHINTDapp(owner, spender, to, value, receiveType);
1179                }
1180            }
1181
1182            return true;
1183        }
1184
1185        function hintMintTo(address to, uint256 amount, string note) onlyOwner public
                returns (bool ret) {
1186            ret = super.hintMintTo(to, amount, note);
1187            postTransfer(0x0, msg.sender, to, amount, HINTReceiver.HINTReceiveType.
                    HINT_MINT);
1188        }
1189
1190        function hintBurnFrom(address from, uint256 value, string note) onlyOwner public
                returns (bool ret) {
1191            ret = super.hintBurnFrom(from, value, note);
1192            postTransfer(0x0, msg.sender, from, value, HINTReceiver.HINTReceiveType.
                    HINT_BURN);
1193        }
1194    }
1195
1196
1197    /**
1198     * @title HINTToken Receiver
1199     */
1200    contract HINTReceiver {
1201        enum HINTReceiveType { HINT_TRANSFER, HINT_MINT, HINT_BURN }
1202        function onHINTReceived(address owner, address spender, uint256 value,
                HINTReceiveType receiveType) public returns (bool);
1203    }
1204
1205    /**
1206     * @title HINTDappSample
1207     */
1208    contract HINTDappSample is HINTReceiver {
1209        event LogOnReceiveHINT(string message, address indexed owner, address indexed
                spender, uint256 value, HINTReceiveType receiveType);
1210
1211        /*@CTK "onHINTReceived correctness"
1212          @post !(__has_overflow)
1213          @post !(__has_buf_overflow)
1214          @post !(__has_assertion_failure)
1215          @post __return == true
```

```
1216      */
1217     function onHINTReceived(address owner, address spender, uint256 value,
             HINTReceiveType receiveType) public returns (bool) {
1218         emit LogOnReceiveHINT("I receive HINT Token.", owner, spender, value,
                 receiveType);
1219         return true;
1220     }
1221 }
```