CertiK Audit Report For HintChain

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

Contents

Disclaimer

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 t
        ) {
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

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

Line 31-42 in File VHC.sol

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
        // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        c = a * b;
        assert(c / a == b);
        return c;
    }
```

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"
      }
    ]

```

```
Function invocation is reverted.
```

Formal Verification Request 2

SafeMath mul

📅 28, May 2019
⏱ 478.15 ms

Line 25-30 in File VHC.sol

```
    /*@CTK "SafeMath mul"
      @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
      @post !__reverted -> c == a * b
      @post !__reverted == !__has_overflow
      @post !(__has_buf_overflow)
     */
```

Line 31-42 in File VHC.sol

```
    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
        // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        c = a * b;
        assert(c / a == b);
        return c;
    }
```

✓ The code meets the specification

Formal Verification Request 3

SafeMath div

📅 28, May 2019
⏱ 8.48 ms

Line 47-53 in File VHC.sol

```
    /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
      @post !(__has_buf_overflow)
      @post !__reverted -> !(__has_assertion_failure)
    */
```

Line 54-59 in File VHC.sol

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

✓ 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

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

Line 71-74 in File VHC.sol

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

✗ 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

```
/*@CTK "SafeMath sub"
  @post (a < b) == __reverted
  @post !__reverted -> __return == a - b
  @post !__reverted -> !__has_overflow
  @post !(__has_buf_overflow)
*/
```

Line 71-74 in File VHC.sol

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

✓ 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

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

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;
}
```

✗ This code violates the specification

```
Counter Example:
Before Execution:
    Input = {
        a = 143
        b = 113
    }
    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"
      }
    ]

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

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

Line 110-112 in File VHC.sol

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

✓ The code meets the specification

Formal Verification Request 9

balanceOf correctness

28, May 2019
6.21 ms

Line 134-136 in File VHC.sol

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

Line 137-139 in File VHC.sol

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

✓ The code meets the specification

Formal Verification Request 10

changeRoot correctness

28, May 2019
49.98 ms

Line 358-365 in File VHC.sol

```
    /*@CTK "changeRoot correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post newRoot != 0x0
      @post __post.root == newRoot
     */
```

Line 366-373 in File VHC.sol

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

        root = newRoot;

        emit ChangedRoot(newRoot);
        return true;
    }
```

✓ The code meets the specification

Formal Verification Request 11

changeSuperOwner correctness

📅 28, May 2019
⏱ 49.6 ms

Line 379-387 in File VHC.sol

```
    /*@CTK "changeSuperOwner correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post msg.sender == root
      @post newSuperOwner != 0x0
      @post __post.superOwner == newSuperOwner
     */
```

Line 388-395 in File VHC.sol

```
    function changeSuperOwner(address newSuperOwner) onlyRoot public returns (bool) {
        require(newSuperOwner != address(0), "This address to be set is zero address(0)
            . Check the input address.");

        superOwner = newSuperOwner;

        emit ChangedSuperOwner(newSuperOwner);
        return true;
    }
```

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 439-448 in File VHC.sol

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

        owners[owner] = true;
        ownerList.push(owner);

        emit AddedNewOwner(owner);
        return true;
    }
```

✓ 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

```
    //@CTK NO_ASF
```

Line 439-448 in File VHC.sol

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

        owners[owner] = true;
        ownerList.push(owner);

        emit AddedNewOwner(owner);
        return true;
    }
```

✓ The code meets the specification

Formal Verification Request 14

newOwner correctness

28, May 2019
19.79 ms

Line 430-438 in File VHC.sol

```
    /*@CTK "newOwner correctness"
      @tag assume_completion
      @post msg.sender == superOwner
      @post owner != 0x0
      @post owners[owner] == false
      @post __post.ownerList.length == ownerList.length + 1
      @post __post.ownerList[ownerList.length] == owner
      @post __post.owners[owner] == true
     */
```

Line 439-448 in File VHC.sol

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

        owners[owner] = true;
        ownerList.push(owner);

        emit AddedNewOwner(owner);
        return true;
    }
```

✓ The code meets the specification

Formal Verification Request 15

lock correctness

📅 28, May 2019
⏱ 29.37 ms

Line 501-508 in File VHC.sol

```
    /*@CTK "lock correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post __post.locked == true
     */
```

Line 509-512 in File VHC.sol

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

✓ The code meets the specification

Formal Verification Request 16

unlock correctness

📅 28, May 2019
⏱ 28.51 ms

Line 514-521 in File VHC.sol

```
    /*@CTK "unlock correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post __post.locked == false
     */
```

Line 522-525 in File VHC.sol

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

✓ The code meets the specification

Formal Verification Request 17

lockTo correctness

📅 28, May 2019
⏱ 90.36 ms

Line 527-536 in File VHC.sol

```
    /*@CTK "lockTo correctness"
      @tag assume_completion
      @pre LOCK_MAX == 255
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post __post.lockValues[addr] == 255
      @post __post.unlockAddrs[addr] == false
     */
```

Line 537-542 in File VHC.sol

```
    function lockTo(address addr, string note) onlyOwner public {
        setLockValue(addr, LOCK_MAX, note);
        unlockAddrs[addr] = false;

        emit LockedTo(addr, true, note);
    }
```

✓ The code meets the specification

Formal Verification Request 18

unlockTo correctness

📅 28, May 2019
⏱ 74.61 ms

Line 544-553 in File VHC.sol

```
    /*@CTK "unlockTo correctness"
      @tag assume_completion
      @pre LOCK_MAX == 255
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post lockValues[addr] == 255 -> __post.lockValues[addr] == 0
      @post __post.unlockAddrs[addr] == true
     */
```

Line 554-560 in File VHC.sol

```
    function unlockTo(address addr, string note) onlyOwner public {
        if (lockValues[addr] == LOCK_MAX)
            setLockValue(addr, 0, note);
        unlockAddrs[addr] = true;

        emit LockedTo(addr, false, note);
    }
```

✓ The code meets the specification

Formal Verification Request 19

setLockValue correctness

📅 28, May 2019
⏱ 2.38 ms

Line 562-569 in File VHC.sol

```
    /*@CTK "setLockValue correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post __post.lockValues[addr] == value
     */
```

Line 570-573 in File VHC.sol

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

✓ The code meets the specification

Formal Verification Request 20

getMyUnlockValue correctness

📅 28, May 2019
⏱ 74.68 ms

Line 578-593 in File VHC.sol

```
    /*@CTK "getMyUnlockValue correctness"
      @tag assume_completion
      @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

```
    //@CTK NO_ASF
```

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 24

delayLock correctness

28, May 2019
42.68 ms

Line 638-644 in File VHC.sol

```
    /*@CTK "delayLock correctness"
      @tag assume_completion
      @post value <= balances[msg.sender]
      @post value < delayLockValues[msg.sender]
           -> __post.delayLockBeforeValues[msg.sender] == delayLockValues[msg.sender]
      @post __post.delayLockValues[msg.sender] == value
     */
```

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 25

If method completes, integer overflow would not happen.

28, May 2019
189.32 ms

Line 665 in File VHC.sol

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

Line 674-676 in File VHC.sol

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

✗ This code violates the specification

```
Counter Example:
Before Execution:
    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": 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 674-676 in File VHC.sol

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

✓ 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

```
    //@CTK NO_ASF
```

Line 674-676 in File VHC.sol

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

✓ The code meets the specification

Formal Verification Request 28

delayUnlock correctness

28, May 2019
38.84 ms

Line 668-673 in File VHC.sol

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

Line 674-676 in File VHC.sol

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

✓ 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

```
    //@CTK NO_OVERFLOW
```

Line 711-726 in File VHC.sol

```
    function getMyUnlockValue() public view returns (uint256) {
        uint256 myUnlockValue;
        address addr = msg.sender;
        if (delayLockTimes[addr] <= now) {
            myUnlockValue = balances[addr].sub(delayLockValues[addr]);
        } else {
            myUnlockValue = balances[addr].sub(delayLockBeforeValues[addr]);
        }

        uint256 superUnlockValue = super.getMyUnlockValue();

        if (myUnlockValue > superUnlockValue)
            return superUnlockValue;
        else
            return myUnlockValue;
    }
```

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 711-726 in File VHC.sol

```
    function getMyUnlockValue() public view returns (uint256) {
        uint256 myUnlockValue;
        address addr = msg.sender;
        if (delayLockTimes[addr] <= now) {
            myUnlockValue = balances[addr].sub(delayLockValues[addr]);
        } else {
            myUnlockValue = balances[addr].sub(delayLockBeforeValues[addr]);
        }

        uint256 superUnlockValue = super.getMyUnlockValue();

        if (myUnlockValue > superUnlockValue)
            return superUnlockValue;
        else
            return myUnlockValue;
    }
```

✓ 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

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

Line 711-726 in File VHC.sol

```
    function getMyUnlockValue() public view returns (uint256) {
        uint256 myUnlockValue;
        address addr = msg.sender;
        if (delayLockTimes[addr] <= now) {
            myUnlockValue = balances[addr].sub(delayLockValues[addr]);
        } else {
            myUnlockValue = balances[addr].sub(delayLockBeforeValues[addr]);
        }

        uint256 superUnlockValue = super.getMyUnlockValue();

        if (myUnlockValue > superUnlockValue)
            return superUnlockValue;
        else
            return myUnlockValue;
    }
```

✗ This code violates the specification

```
Counter Example:
Before Execution:
    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 = 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

```
/*@CTK "DelayLockableToken getMyUnlockValue correctness"
  @tag assume_completion
  @pre balances[msg.sender] >= delayLockValues[msg.sender]
  @pre balances[msg.sender] >= delayLockBeforeValues[msg.sender]
  @post locked && !unlockAddrs[msg.sender]
       -> __return == 0
  @post !locked && (balances[msg.sender] <= lockValues[msg.sender])
       -> __return == 0
  @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] <= lockValues[
      msg.sender])
       -> __return == 0
  @post !locked && (balances[msg.sender] > lockValues[msg.sender])
       && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
           delayLockValues[msg.sender]
       && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
           delayLockBeforeValues[msg.sender]
       -> __return == balances[msg.sender] - lockValues[msg.sender]
  @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] > lockValues[
      msg.sender])
       && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
           delayLockValues[msg.sender]
       && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
           delayLockBeforeValues[msg.sender]
       -> __return == balances[msg.sender] - lockValues[msg.sender]
  @post !locked && (balances[msg.sender] > lockValues[msg.sender])
       && balances[msg.sender] - lockValues[msg.sender] >= balances[msg.sender] -
           delayLockValues[msg.sender]
       && delayLockTimes[msg.sender] <= now
       -> __return == balances[msg.sender] - delayLockValues[msg.sender]
  @post !locked && (balances[msg.sender] > lockValues[msg.sender])
       && balances[msg.sender] - lockValues[msg.sender] >= balances[msg.sender] -
           delayLockBeforeValues[msg.sender]
       && delayLockTimes[msg.sender] > now
       -> __return == balances[msg.sender] - delayLockBeforeValues[msg.sender]
 */
```

Line 711-726 in File VHC.sol

```
function getMyUnlockValue() public view returns (uint256) {
    uint256 myUnlockValue;
    address addr = msg.sender;
    if (delayLockTimes[addr] <= now) {
        myUnlockValue = balances[addr].sub(delayLockValues[addr]);
    } else {
        myUnlockValue = balances[addr].sub(delayLockBeforeValues[addr]);
    }

    uint256 superUnlockValue = super.getMyUnlockValue();

    if (myUnlockValue > superUnlockValue)
        return superUnlockValue;
    else
        return myUnlockValue;
}
```

✅ 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

```
    //@CTK NO_OVERFLOW
```

Line 777-779 in File VHC.sol

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

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 777-779 in File VHC.sol

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

✓ The code meets the specification

Formal Verification Request 35

transfer correctness

📅 28, May 2019
⏱ 33937.85 ms

Line 764-776 in File VHC.sol

```
    /*@CTK "transfer correctness"
      @tag assume_completion
      @post to != 0x0
      @post to != address(this)
      @post value <= balances[msg.sender]
      @post (value <= balances[msg.sender] - delayLockValues[msg.sender])
            || (value <= balances[msg.sender] - delayLockBeforeValues[msg.sender])
      @post value <= balances[msg.sender] - lockValues[msg.sender]
      @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

```
/*@CTK "transferFrom correctness"
  @tag assume_completion
  @post to != 0x0
  @post to != address(this)
  @post value <= balances[from] && value <= allowed[from][msg.sender]
  @post (value <= balances[from] - delayLockValues[from])
      || (value <= balances[from] - delayLockBeforeValues[from])
  @post value <= balances[from] - lockValues[from]
  @post (!locked || unlockAddrs[from])
  @post to != from -> __post.balances[from] == balances[from] - value
  @post to != from -> __post.balances[to] == balances[to] + value
  @post to == from -> __post.balances[from] == balances[from]
  @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
 */
```

Line 805-807 in File VHC.sol

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

✓ The code meets the specification

Formal Verification Request 39

approve correctness

28, May 2019
96.92 ms

Line 816-821 in File VHC.sol

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

Line 822-824 in File VHC.sol

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

✓ The code meets the specification

Formal Verification Request 40

increaseAllowance

📅 28, May 2019
⏱ 184.07 ms

Line 831-838 in File VHC.sol

```
/*@CTK increaseAllowance
  @tag assume_completion
  @post !(__has_overflow)
  @post !(__has_buf_overflow)
  @post !(__has_assertion_failure)
  @post __post.allowed[msg.sender][spender] ==
       allowed[msg.sender][spender] + addedValue
 */
```

Line 839-841 in File VHC.sol

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

✅ The code meets the specification

Formal Verification Request 41

decreaseAllowance

📅 28, May 2019
⏱ 230.86 ms

Line 848-857 in File VHC.sol

```
/*@CTK decreaseAllowance
  @tag assume_completion
  @post !(__has_overflow)
  @post !(__has_buf_overflow)
  @post !(__has_assertion_failure)
  @post allowed[msg.sender][spender] >= subtractedValue ->
       __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] -
           subtractedValue
  @post allowed[msg.sender][spender] < subtractedValue ->
       __post.allowed[msg.sender][spender] == 0
 */
```

Line 858-860 in File VHC.sol

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

✅ The code meets the specification

Formal Verification Request 42

hintMintTo

📅 28, May 2019
⏱ 379.11 ms

Line 880-890 in File VHC.sol

```
    /*@CTK hintMintTo
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post to != address(0)
      @post owners[msg.sender] == true
      @post __post.totalSupply_ == totalSupply_ + amount
      @post __post.balances[to] == balances[to] + amount
      @post ret == true
     */
```

Line 891-894 in File VHC.sol

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

✅ The code meets the specification

Formal Verification Request 43

hintBurnFrom

📅 28, May 2019
⏱ 438.72 ms

Line 909-920 in File VHC.sol

```
    /*@CTK hintBurnFrom
      @tag assume_completion
      @pre balances[from] <= totalSupply_
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post value <= balances[from]
      @post __post.totalSupply_ == totalSupply_ - value
      @post __post.balances[from] == balances[from] - value
      @post ret == true
     */
```

Line 921-924 in File VHC.sol

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

✓ The code meets the specification

Formal Verification Request 44

hintBurnWhenMoveToMainnet

28, May 2019
537.63 ms

Line 929-940 in File VHC.sol

```
    /*@CTK hintBurnWhenMoveToMainnet
      @tag assume_completion
      @pre balances[burner] <= totalSupply_
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post value <= balances[burner]
      @post __post.totalSupply_ == totalSupply_ - value
      @post __post.balances[burner] == balances[burner] - value
      @post ret == true
     */
```

Line 941-944 in File VHC.sol

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

✓ 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

```
    //@CTK NO_OVERFLOW
```

Line 977-987 in File VHC.sol

```
    function hintSell(
        address from,
        address to,
        uint256 value,
        string note
    ) onlyOwner public returns (bool ret) {
        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
```

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

Line 977-987 in File VHC.sol

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

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

✓ 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

```
  //@CTK NO_OVERFLOW
```

Line 1035-1045 in File VHC.sol

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

      ret = hintTransferFrom(from, to, value, note);
      emit HINTTransferToTeam(from, msg.sender, to, value, note);
  }
```

✓ 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

```
    //@CTK NO_BUF_OVERFLOW
```

Line 1035-1045 in File VHC.sol

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

        ret = hintTransferFrom(from, to, value, note);
        emit HINTTransferToTeam(from, msg.sender, to, value, note);
    }
```

✓ The code meets the specification

Formal Verification Request 50

hintTransferToTeam correctness

28, May 2019
43971.24 ms

Line 1020-1034 in File VHC.sol

```
    /*@CTK "hintTransferToTeam correctness"
      @tag assume_completion
      @post to != 0x0
      @post to != address(this)
      @post owners[msg.sender] == true
      @post value <= balances[from] && value <= allowed[from][msg.sender]
      @post (value <= balances[from] - delayLockValues[from])
            || (value <= balances[from] - delayLockBeforeValues[from])
      @post value <= balances[from] - lockValues[from]
      @post (!locked || unlockAddrs[from])
      @post to != from -> __post.balances[from] == balances[from] - value
      @post to != from -> __post.balances[to] == balances[to] + value
      @post to == from -> __post.balances[from] == balances[from]
      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
     */
```

Line 1035-1045 in File VHC.sol

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

        ret = hintTransferFrom(from, to, value, note);
        emit HINTTransferToTeam(from, msg.sender, to, value, note);
    }
```

✓ 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

```
//@CTK NO_OVERFLOW
```

Line 1068-1078 in File VHC.sol

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

    ret = hintTransferFrom(from, to, value, note);
    emit HINTTransferToPartner(from, msg.sender, to, value, note);
}
```

✓ 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

```
//@CTK NO_BUF_OVERFLOW
```

Line 1068-1078 in File VHC.sol

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

    ret = hintTransferFrom(from, to, value, note);
    emit HINTTransferToPartner(from, msg.sender, to, value, note);
}
```

✓ The code meets the specification

Formal Verification Request 53

hintTransferToPartner correctness

📅 28, May 2019
⏱ 40296.02 ms

Line 1053-1067 in File VHC.sol

```
    /*@CTK "hintTransferToPartner correctness"
      @tag assume_completion
      @post to != 0x0
      @post to != address(this)
      @post owners[msg.sender] == true
      @post value <= balances[from] && value <= allowed[from][msg.sender]
      @post (value <= balances[from] - delayLockValues[from])
            || (value <= balances[from] - delayLockBeforeValues[from])
      @post value <= balances[from] - lockValues[from]
      @post (!locked || unlockAddrs[from])
      @post to != from -> __post.balances[from] == balances[from] - value
      @post to != from -> __post.balances[to] == balances[to] + value
      @post to == from -> __post.balances[from] == balances[from]
      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
     */
```

Line 1068-1078 in File VHC.sol

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

        ret = hintTransferFrom(from, to, value, note);
        emit HINTTransferToPartner(from, msg.sender, to, value, note);
    }
```

✅ The code meets the specification

Formal Verification Request 54

constructor correctness

📅 28, May 2019
⏱ 24.33 ms

Line 1150-1157 in File VHC.sol

```
    /*@CTK "constructor correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post __post.totalSupply_ == INITIAL_SUPPLY
      @post __post.balances[msg.sender] == __post.totalSupply_
```

```
 */
```

Line 1158-1162 in File VHC.sol

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

✓ The code meets the specification

Formal Verification Request 55

onHINTReceived correctness

28, May 2019
8.04 ms

Line 1211-1216 in File VHC.sol

```
/*@CTK "onHINTReceived correctness"
  @post !(__has_overflow)
  @post !(__has_buf_overflow)
  @post !(__has_assertion_failure)
  @post __return == true
 */
```

Line 1217-1220 in File VHC.sol

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

✓ The code meets the specification

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File VHC.sol

```
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

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

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 649 in File VHC.sol

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

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 651 in File VHC.sol

```
            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

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

"now" can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 714 in File VHC.sol

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

"now" can be influenced by minors to some degree

Source Code with CertiK Labels

File VHC.sol

```
pragma solidity ^0.4.24;

/**
 * @title ERC20Basic
 * dev Simpler version of ERC20 interface
 * See https://github.com/ethereum/EIPs/issues/179
 */
contract ERC20Basic {
    function totalSupply() public view returns (uint256);
    function balanceOf(address who) public view returns (uint256);
    function transfer(address to, uint256 value) public returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
}

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

    /**
    * dev Multiplies two numbers, throws on overflow.
    */
    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath mul"
      @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
      @post !__reverted -> c == a * b
      @post !__reverted == !__has_overflow
      @post !(__has_buf_overflow)
     */
    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
        // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if (a == 0) {
            return 0;
        }

        c = a * b;
        assert(c / a == b);
        return c;
    }

    /**
    * dev Integer division of two numbers, truncating the quotient.
    */
    /*@CTK "SafeMath div"
      @post b != 0 -> !__reverted
      @post !__reverted -> __return == a / b
      @post !__reverted -> !__has_overflow
      @post !(__has_buf_overflow)
      @post !__reverted -> !(__has_assertion_failure)
    */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        // assert(b > 0); // Solidity automatically throws when dividing by 0
        // uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold
        return a / b;
    }

    /**
    * dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater
        than minuend).
    */
    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath sub"
      @post (a < b) == __reverted
      @post !__reverted -> __return == a - b
      @post !__reverted -> !__has_overflow
      @post !(__has_buf_overflow)
    */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }

    /**
    * dev Adds two numbers, throws on overflow.
    */
    //@CTK FAIL NO_ASF
    /*@CTK "SafeMath add"
      @post (a + b < a || a + b < b) == __reverted
      @post !__reverted -> c == a + b
      @post !__reverted -> !__has_overflow
      @post !(__has_buf_overflow)
    */
    function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
        c = a + b;
        assert(c >= a);
        return c;
    }
}

/**
 * @title Basic token
 * dev Basic version of StandardToken, with no allowances.
 */
contract BasicToken is ERC20Basic {
    using SafeMath for uint256;

    mapping(address => uint256) balances;

    uint256 totalSupply_;

    /**
    * dev Total number of tokens in existence
    */
    /*@CTK "totalSupply correctness"
      @post __return == totalSupply_
     */
    function totalSupply() public view returns (uint256) {
        return totalSupply_;
    }

    /**
    * dev Transfer token for a specified address
    * @param _to The address to transfer to.
    * @param _value The amount to be transferred.
    */
    function transfer(address _to, uint256 _value) public returns (bool) {
        require(_to != address(0), "Recipient address is zero address(0). Check the
            address again.");
        require(_value <= balances[msg.sender], "The balance of account is insufficient
            .");

        balances[msg.sender] = balances[msg.sender].sub(_value);
        balances[_to] = balances[_to].add(_value);
        emit Transfer(msg.sender, _to, _value);
        return true;
    }

    /**
    * dev Gets the balance of the specified address.
    * @param _owner The address to query the the balance of.
    * @return An uint256 representing the amount owned by the passed address.
    */
    /*@CTK "balanceOf correctness"
      @post __return == balances[_owner]
     */
    function balanceOf(address _owner) public view returns (uint256) {
        return balances[_owner];
    }

}

/**
 * @title ERC20 interface
 * dev see https://github.com/ethereum/EIPs/issues/20
 */
contract ERC20 is ERC20Basic {
    function allowance(address owner, address spender)
    public view returns (uint256);

    function transferFrom(address from, address to, uint256 value)
    public returns (bool);

    function approve(address spender, uint256 value) public returns (bool);
    event Approval(
        address indexed owner,
        address indexed spender,
        uint256 value
    );
}

/**
 * @title Standard ERC20 token
 *
 * dev Implementation of the basic standard token.
 * https://github.com/ethereum/EIPs/issues/20
 * Based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master/
    smart_contract/FirstBloodToken.sol
 */
contract StandardToken is ERC20, BasicToken {

    mapping (address => mapping (address => uint256)) internal allowed;


    /**
     * dev Transfer tokens from one address to another
     * @param _from address The address which you want to send tokens from
     * @param _to address The address which you want to transfer to
     * @param _value uint256 the amount of tokens to be transferred
     */
    function transferFrom(
        address _from,
        address _to,
        uint256 _value
    )
    public
    returns (bool)
    {
        require(_to != address(0), "Recipient address is zero address(0). Check the
            address again.");
        require(_value <= balances[_from], "The balance of account is insufficient.");
        require(_value <= allowed[_from][msg.sender], "Insufficient tokens approved
            from account owner.");

        balances[_from] = balances[_from].sub(_value);
        balances[_to] = balances[_to].add(_value);
        allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
        emit Transfer(_from, _to, _value);
        return true;
    }

    /**
     * dev Approve the passed address to spend the specified amount of tokens on
        behalf of msg.sender.
     * Beware that changing an allowance with this method brings the risk that someone
         may use both the old
     * and the new allowance by unfortunate transaction ordering. One possible
        solution to mitigate this
     * race condition is to first reduce the spender's allowance to 0 and set the
        desired value afterwards:
     * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
     * @param _spender The address which will spend the funds.
     * @param _value The amount of tokens to be spent.
     */
    function approve(address _spender, uint256 _value) public returns (bool) {
        allowed[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value);
        return true;
    }

    /**
     * dev Function to check the amount of tokens that an owner allowed to a spender.
     * @param _owner address The address which owns the funds.
     * @param _spender address The address which will spend the funds.
```

```
     * @return A uint256 specifying the amount of tokens still available for the
         spender.
     */
    function allowance(
        address _owner,
        address _spender
    )
    public
    view
    returns (uint256)
    {
        return allowed[_owner][_spender];
    }

    /**
     * dev Increase the amount of tokens that an owner allowed to a spender.
     * approve should be called when allowed[_spender] == 0. To increment
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * @param _spender The address which will spend the funds.
     * @param _addedValue The amount of tokens to increase the allowance by.
     */
    function increaseApproval(
        address _spender,
        uint256 _addedValue
    )
    public
    returns (bool)
    {
        allowed[msg.sender][_spender] = (
        allowed[msg.sender][_spender].add(_addedValue));
        emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
        return true;
    }

    /**
     * dev Decrease the amount of tokens that an owner allowed to a spender.
     * approve should be called when allowed[_spender] == 0. To decrement
     * allowed value is better to use this function to avoid 2 calls (and wait until
     * the first transaction is mined)
     * From MonolithDAO Token.sol
     * @param _spender The address which will spend the funds.
     * @param _subtractedValue The amount of tokens to decrease the allowance by.
     */
    function decreaseApproval(
        address _spender,
        uint256 _subtractedValue
    )
    public
    returns (bool)
    {
        uint256 oldValue = allowed[msg.sender][_spender];
        if (_subtractedValue > oldValue) {
            allowed[msg.sender][_spender] = 0;
        } else {
            allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
        }
        emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
        return true;
    }

}

/**
 * Utility library of inline functions on addresses
 */
library AddressUtils {

    /**
     * Returns whether the target address is a contract
     * dev This function will return false if invoked during the constructor of a contract,
     * as the code is not actually created until after the constructor finishes.
     * @param addr address to check
     * @return whether the target address is a contract
     */
    function isContract(address addr) internal view returns (bool) {
        uint256 size;
        // XXX Currently there is no better way to check if there is a contract in an address
        // than to check the size of the code at that address.
        // See https://ethereum.stackexchange.com/a/14016/36603
        // for more details about how this works.
        // TODO Check this again before the Serenity release, because all addresses will be
        // contracts then.
        // solium-disable-next-line security/no-inline-assembly
        assembly { size := extcodesize(addr) }
        return size > 0;
    }

}

/**
 * @title MultiOwnable
 * dev
 */
contract MultiOwnable {
    using SafeMath for uint256;

    address public root; //                       superOwner         .
                                                  .
    address public superOwner;
    mapping (address => bool) public owners;
    address[] public ownerList;

    // for changeSuperOwnerByDAO
    // mapping(address => mapping (address => bool)) public preSuperOwnerMap;
    mapping(address => address) public candidateSuperOwnerMap;


    event ChangedRoot(address newRoot);
    event ChangedSuperOwner(address newSuperOwner);
    event AddedNewOwner(address newOwner);
    event DeletedOwner(address deletedOwner);

    constructor() public {
        root = msg.sender;
        superOwner = msg.sender;
        owners[root] = true;

        ownerList.push(msg.sender);

    }

    modifier onlyRoot() {
        require(msg.sender == root, "Root privilege is required.");
        _;
    }

    modifier onlySuperOwner() {
        require(msg.sender == superOwner, "SuperOwner priviledge is required.");
        _;
    }

    modifier onlyOwner() {
        require(owners[msg.sender], "Owner priviledge is required.");
        _;
    }

    /**
     * dev root      (root    root    superOwner
         .)
     * dev                                            ,
                                                 !
     */
    /*@CTK "changeRoot correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post newRoot != 0x0
      @post __post.root == newRoot
     */
    function changeRoot(address newRoot) onlyRoot public returns (bool) {
        require(newRoot != address(0), "This address to be set is zero address(0).
            Check the input address.");

        root = newRoot;

        emit ChangedRoot(newRoot);
        return true;
    }

    /**
     * dev superOwner      (root    root    superOwner
              .)
     * dev      superOwner                                ,     superOwner
                                                            !
     */
    /*@CTK "changeSuperOwner correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post msg.sender == root
      @post newSuperOwner != 0x0
      @post __post.superOwner == newSuperOwner
     */
    function changeSuperOwner(address newSuperOwner) onlyRoot public returns (bool) {
        require(newSuperOwner != address(0), "This address to be set is zero address(0). Check the input address.");

        superOwner = newSuperOwner;

        emit ChangedSuperOwner(newSuperOwner);
        return true;
    }

    /**
     * dev owner      1/2                     superOwner                         .
     */
    function changeSuperOwnerByDAO(address newSuperOwner) onlyOwner public returns (bool) {
        require(newSuperOwner != address(0), "This address to be set is zero address(0). Check the input address.");
        require(newSuperOwner != candidateSuperOwnerMap[msg.sender], "You have already voted for this account.");

        candidateSuperOwnerMap[msg.sender] = newSuperOwner;

        uint8 votingNumForSuperOwner = 0;
        uint8 i = 0;

        for (i = 0; i < ownerList.length; i++) {
            if (candidateSuperOwnerMap[ownerList[i]] == newSuperOwner)
                votingNumForSuperOwner++;
        }

        if (votingNumForSuperOwner > ownerList.length / 2) { //                 DAO          => superOwner
            superOwner = newSuperOwner;

            //
            for (i = 0; i < ownerList.length; i++) {
                delete candidateSuperOwnerMap[ownerList[i]];
            }

            emit ChangedSuperOwner(newSuperOwner);
        }

        return true;
    }

    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "newOwner correctness"
      @tag assume_completion
      @post msg.sender == superOwner
      @post owner != 0x0
      @post owners[owner] == false
      @post __post.ownerList.length == ownerList.length + 1
      @post __post.ownerList[ownerList.length] == owner
      @post __post.owners[owner] == true
     */
    function newOwner(address owner) onlySuperOwner public returns (bool) {
        require(owner != address(0), "This address to be set is zero address(0). Check the input address.");
        require(!owners[owner], "This address is already registered.");

        owners[owner] = true;
        ownerList.push(owner);

        emit AddedNewOwner(owner);
        return true;
    }

    function deleteOwner(address owner) onlySuperOwner public returns (bool) {
        require(owners[owner], "This input address is not a super owner.");
        delete owners[owner];

        for (uint256 i = 0; i < ownerList.length; i++) {
            if (ownerList[i] == owner) {
                ownerList[i] = ownerList[ownerList.length.sub(1)];
                ownerList.length = ownerList.length.sub(1);
                break;
            }
        }

        emit DeletedOwner(owner);
        return true;
    }
}

/**
 * @title Lockable token
 */
contract LockableToken is StandardToken, MultiOwnable {
    bool public locked = true;
    uint256 public constant LOCK_MAX = uint256(-1);

    /**
     * dev
     */
    mapping(address => bool) public unlockAddrs;

    /**
     * dev                lock value
     * dev -       0         :          0                                        .
     * dev -       LOCK_MAX        :          uint256                                        .
     */
    mapping(address => uint256) public lockValues;

    event Locked(bool locked, string note);
    event LockedTo(address indexed addr, bool locked, string note);
    event SetLockValue(address indexed addr, uint256 value, string note);

    constructor() public {
        unlockTo(msg.sender, "");
    }

    modifier checkUnlock (address addr, uint256 value) {
        require(!locked || unlockAddrs[addr], "The account is currently locked.");
        require(balances[addr] >= value, "Transferable limit exceeded. Check the status
             of the lock value.");
        require(balances[addr] - value >= lockValues[addr], "Transferable limit
            exceeded. Check the status of the lock value.");
        _;
    }

    /*@CTK "lock correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post __post.locked == true
     */
    function lock(string note) onlyOwner public {
        locked = true;
        emit Locked(locked, note);
    }

    /*@CTK "unlock correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post __post.locked == false
     */
    function unlock(string note) onlyOwner public {
        locked = false;
        emit Locked(locked, note);
    }

    /*@CTK "lockTo correctness"
      @tag assume_completion
      @pre LOCK_MAX == 255
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post __post.lockValues[addr] == 255
      @post __post.unlockAddrs[addr] == false
     */
    function lockTo(address addr, string note) onlyOwner public {
        setLockValue(addr, LOCK_MAX, note);
        unlockAddrs[addr] = false;

        emit LockedTo(addr, true, note);
    }

    /*@CTK "unlockTo correctness"
      @tag assume_completion
      @pre LOCK_MAX == 255
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post lockValues[addr] == 255 -> __post.lockValues[addr] == 0
      @post __post.unlockAddrs[addr] == true
     */
    function unlockTo(address addr, string note) onlyOwner public {
        if (lockValues[addr] == LOCK_MAX)
            setLockValue(addr, 0, note);
        unlockAddrs[addr] = true;

        emit LockedTo(addr, false, note);
    }

    /*@CTK "setLockValue correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post __post.lockValues[addr] == value
     */
    function setLockValue(address addr, uint256 value, string note) onlyOwner public {
        lockValues[addr] = value;
        emit SetLockValue(addr, value, note);
    }

    /**
     * dev                                      .
     */
    /*@CTK "getMyUnlockValue correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post locked && !unlockAddrs[msg.sender]
            -> __return == 0
      @post !locked && (balances[msg.sender] > lockValues[msg.sender])
            -> __return == balances[msg.sender] - lockValues[msg.sender]
      @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] > lockValues[
          msg.sender])
            -> __return == balances[msg.sender] - lockValues[msg.sender]
      @post !locked && (balances[msg.sender] <= lockValues[msg.sender])
            -> __return == 0
      @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] <= lockValues[
          msg.sender])
            -> __return == 0
     */
    function getMyUnlockValue() public view returns (uint256) {
        address addr = msg.sender;
        if ((!locked || unlockAddrs[addr]) && balances[addr] > lockValues[addr])
            return balances[addr].sub(lockValues[addr]);
        else
            return 0;
    }

    function transfer(address to, uint256 value) checkUnlock(msg.sender, value) public
         returns (bool) {
        return super.transfer(to, value);
    }

    function transferFrom(address from, address to, uint256 value) checkUnlock(from,
        value) public returns (bool) {
        return super.transferFrom(from, to, value);
    }
}

/**
 * @title DelayLockableToken
 * dev                                       lock                    .
                                       12                            .
 */
contract DelayLockableToken is LockableToken {
    mapping(address => uint256) public delayLockValues;
    mapping(address => uint256) public delayLockBeforeValues;
    mapping(address => uint256) public delayLockTimes;

    event SetDelayLockValue(address indexed addr, uint256 value, uint256 time);

    modifier checkDelayUnlock (address addr, uint256 value) {
        require(balances[addr] >= value);
        if (delayLockTimes[msg.sender] <= now) {
            require (balances[addr] - value >= delayLockValues[addr], "Transferable
                limit exceeded. Change the balance lock value first and then use it");
        } else {
            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.");
        }
        _;
    }

    /**
     * dev                                   .                                    ,
                         12                          .
     */
    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "delayLock correctness"
      @tag assume_completion
      @post value <= balances[msg.sender]
      @post value < delayLockValues[msg.sender]
            -> __post.delayLockBeforeValues[msg.sender] == delayLockValues[msg.sender]
      @post __post.delayLockValues[msg.sender] == value
     */
    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;
    }

    /**
     * dev                                         .
     */
    //@CTK FAIL NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "delayUnlock correctness"
      @tag assume_completion
      @post delayLockValues[msg.sender] != 0
            -> __post.delayLockBeforeValues[msg.sender] == delayLockValues[msg.sender]
      @post __post.delayLockValues[msg.sender] == 0
     */
    function delayUnlock() public returns (bool) {
        return delayLock(0);
    }

    /**
     * dev                                  .
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK FAIL NO_ASF
    /*@CTK "DelayLockableToken getMyUnlockValue correctness"
      @tag assume_completion
      @pre balances[msg.sender] >= delayLockValues[msg.sender]
      @pre balances[msg.sender] >= delayLockBeforeValues[msg.sender]
      @post locked && !unlockAddrs[msg.sender]
            -> __return == 0
      @post !locked && (balances[msg.sender] <= lockValues[msg.sender])
            -> __return == 0
      @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] <= lockValues[
          msg.sender])
            -> __return == 0
      @post !locked && (balances[msg.sender] > lockValues[msg.sender])
            && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                delayLockValues[msg.sender]
            && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                delayLockBeforeValues[msg.sender]
            -> __return == balances[msg.sender] - lockValues[msg.sender]
      @post locked && unlockAddrs[msg.sender] && (balances[msg.sender] > lockValues[
          msg.sender])
            && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                delayLockValues[msg.sender]
            && balances[msg.sender] - lockValues[msg.sender] < balances[msg.sender] -
                delayLockBeforeValues[msg.sender]
            -> __return == balances[msg.sender] - lockValues[msg.sender]
      @post !locked && (balances[msg.sender] > lockValues[msg.sender])
```

```
            && balances[msg.sender] - lockValues[msg.sender] >= balances[msg.sender] -
                delayLockValues[msg.sender]
            && delayLockTimes[msg.sender] <= now
            -> __return == balances[msg.sender] - delayLockValues[msg.sender]
      @post !locked && (balances[msg.sender] > lockValues[msg.sender])
            && balances[msg.sender] - lockValues[msg.sender] >= balances[msg.sender] -
                delayLockBeforeValues[msg.sender]
            && delayLockTimes[msg.sender] > now
            -> __return == balances[msg.sender] - delayLockBeforeValues[msg.sender]
     */
    function getMyUnlockValue() public view returns (uint256) {
        uint256 myUnlockValue;
        address addr = msg.sender;
        if (delayLockTimes[addr] <= now) {
            myUnlockValue = balances[addr].sub(delayLockValues[addr]);
        } else {
            myUnlockValue = balances[addr].sub(delayLockBeforeValues[addr]);
        }

        uint256 superUnlockValue = super.getMyUnlockValue();

        if (myUnlockValue > superUnlockValue)
            return superUnlockValue;
        else
            return myUnlockValue;
    }

    function transfer(address to, uint256 value) checkDelayUnlock(msg.sender, value)
        public returns (bool) {
        return super.transfer(to, value);
    }

    function transferFrom(address from, address to, uint256 value) checkDelayUnlock(
        from, value) public returns (bool) {
        return super.transferFrom(from, to, value);
    }
}

/**
 * @title HINTBaseToken
 * dev                                                   .
 */
contract HINTBaseToken is DelayLockableToken {
    event HINTTransfer(address indexed from, address indexed to, uint256 value, string
        note);
    event HINTTransferFrom(address indexed owner, address indexed spender, address
        indexed to, uint256 value, string note);
    event HINTApproval(address indexed owner, address indexed spender, uint256 value,
        string note);

    event HINTMintTo(address indexed controller, address indexed to, uint256 amount,
        string note);
    event HINTBurnFrom(address indexed controller, address indexed from, uint256 value
        , string note);

    event HINTBurnWhenMoveToMainnet(address indexed controller, address indexed from,
        uint256 value, string note);

    event HINTSell(address indexed owner, address indexed spender, address indexed to, uint256 value, string note);
    event HINTSellByOtherCoin(address indexed owner, address indexed spender, address indexed to, uint256 value, uint256 processIdHash, uint256 userIdHash, string note);

    event HINTTransferToTeam(address indexed owner, address indexed spender, address indexed to, uint256 value, string note);
    event HINTTransferToPartner(address indexed owner, address indexed spender, address indexed to, uint256 value, string note);

    event HINTTransferToEcosystem(address indexed owner, address indexed spender, address indexed to, uint256 value, uint256 processIdHash, uint256 userIdHash, string note);
    event HINTTransferToBounty(address indexed owner, address indexed spender, address indexed to, uint256 value, uint256 processIdHash, uint256 userIdHash, string note);

    // ERC20                                super                                   hint~
                                      .
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //CTK FAIL NO_ASF
    /*@CTK "transfer correctness"
      @tag assume_completion
      @post to != 0x0
      @post to != address(this)
      @post value <= balances[msg.sender]
      @post (value <= balances[msg.sender] - delayLockValues[msg.sender])
            || (value <= balances[msg.sender] - delayLockBeforeValues[msg.sender])
      @post value <= balances[msg.sender] - lockValues[msg.sender]
      @post (!locked || unlockAddrs[msg.sender])
      @post to != msg.sender -> __post.balances[msg.sender] == balances[msg.sender] - value
      @post to != msg.sender -> __post.balances[to] == balances[to] + value
      @post to == msg.sender -> __post.balances[msg.sender] == balances[msg.sender]
     */
    function transfer(address to, uint256 value) public returns (bool ret) {
        return hintTransfer(to, value, "");
    }

    function hintTransfer(address to, uint256 value, string note) public returns (bool ret) {
        require(to != address(this), "The receive address is the Contact Address of HINTToken. You cannot send money to this address.");

        ret = super.transfer(to, value);
        emit HINTTransfer(msg.sender, to, value, note);
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //CTK FAIL NO_ASF
    /*@CTK "transferFrom correctness"
      @tag assume_completion
      @post to != 0x0
      @post to != address(this)
      @post value <= balances[from] && value <= allowed[from][msg.sender]
      @post (value <= balances[from] - delayLockValues[from])
           || (value <= balances[from] - delayLockBeforeValues[from])
      @post value <= balances[from] - lockValues[from]
      @post (!locked || unlockAddrs[from])
      @post to != from -> __post.balances[from] == balances[from] - value
      @post to != from -> __post.balances[to] == balances[to] + value
      @post to == from -> __post.balances[from] == balances[from]
      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
     */
    function transferFrom(address from, address to, uint256 value) public returns (
        bool) {
        return hintTransferFrom(from, to, value, "");
    }

    function hintTransferFrom(address from, address to, uint256 value, string note)
        public returns (bool ret) {
        require(to != address(this), "The receive address is the Contact Address of
            HINTToken. You cannot send money to this address.");

        ret = super.transferFrom(from, to, value);
        emit HINTTransferFrom(from, msg.sender, to, value, note);
    }

    /*@CTK "approve correctness"
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post __post.allowed[msg.sender][spender] == value
     */
    function approve(address spender, uint256 value) public returns (bool) {
        return hintApprove(spender, value, "");
    }

    function hintApprove(address spender, uint256 value, string note) public returns (
        bool ret) {
        ret = super.approve(spender, value);
        emit HINTApproval(msg.sender, spender, value, note);
    }

    /*@CTK increaseAllowance
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post __post.allowed[msg.sender][spender] ==
           allowed[msg.sender][spender] + addedValue
     */
    function increaseApproval(address spender, uint256 addedValue) public returns (
        bool) {
        return hintIncreaseApproval(spender, addedValue, "");
    }

    function hintIncreaseApproval(address spender, uint256 addedValue, string note)
        public returns (bool ret) {
        ret = super.increaseApproval(spender, addedValue);
        emit HINTApproval(msg.sender, spender, allowed[msg.sender][spender], note);
    }

```

```
/*@CTK decreaseAllowance
  @tag assume_completion
  @post !(__has_overflow)
  @post !(__has_buf_overflow)
  @post !(__has_assertion_failure)
  @post allowed[msg.sender][spender] >= subtractedValue ->
        __post.allowed[msg.sender][spender] == allowed[msg.sender][spender] - subtractedValue
  @post allowed[msg.sender][spender] < subtractedValue ->
        __post.allowed[msg.sender][spender] == 0
 */
function decreaseApproval(address spender, uint256 subtractedValue) public returns (bool) {
    return hintDecreaseApproval(spender, subtractedValue, "");
}

function hintDecreaseApproval(address spender, uint256 subtractedValue, string note) public returns (bool ret) {
    ret = super.decreaseApproval(spender, subtractedValue);
    emit HINTApproval(msg.sender, spender, allowed[msg.sender][spender], note);
}

/**
 * dev                   .                                     .
 */
function mintTo(address to, uint256 amount) internal returns (bool) {
    require(to != address(0x0), "This address to be set is zero address(0). Check the input address.");

    totalSupply_ = totalSupply_.add(amount);
    balances[to] = balances[to].add(amount);

    emit Transfer(address(0), to, amount);
    return true;
}

/*@CTK hintMintTo
  @tag assume_completion
  @post !(__has_overflow)
  @post !(__has_buf_overflow)
  @post !(__has_assertion_failure)
  @post to != address(0)
  @post owners[msg.sender] == true
  @post __post.totalSupply_ == totalSupply_ + amount
  @post __post.balances[to] == balances[to] + amount
  @post ret == true
 */
function hintMintTo(address to, uint256 amount, string note) onlyOwner public returns (bool ret) {
    ret = mintTo(to, amount);
    emit HINTMintTo(msg.sender, to, amount, note);
}

/**
 * dev             .                                     .
 */
function burnFrom(address from, uint256 value) internal returns (bool) {
    require(value <= balances[from], "Your balance is insufficient.");
```

```

        balances[from] = balances[from].sub(value);
        totalSupply_ = totalSupply_.sub(value);

        emit Transfer(from, address(0), value);
        return true;
    }

    /*@CTK hintBurnFrom
      @tag assume_completion
      @pre balances[from] <= totalSupply_
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post value <= balances[from]
      @post __post.totalSupply_ == totalSupply_ - value
      @post __post.balances[from] == balances[from] - value
      @post ret == true
     */
    function hintBurnFrom(address from, uint256 value, string note) onlyOwner public returns (bool ret) {
        ret = burnFrom(from, value);
        emit HINTBurnFrom(msg.sender, from, value, note);
    }

    /**
     * dev                                    .
     */
    /*@CTK hintBurnWhenMoveToMainnet
      @tag assume_completion
      @pre balances[burner] <= totalSupply_
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post owners[msg.sender] == true
      @post value <= balances[burner]
      @post __post.totalSupply_ == totalSupply_ - value
      @post __post.balances[burner] == balances[burner] - value
      @post ret == true
     */
    function hintBurnWhenMoveToMainnet(address burner, uint256 value, string note) onlyOwner public returns (bool ret) {
        ret = hintBurnFrom(burner, value, note);
        emit HINTBurnWhenMoveToMainnet(msg.sender, burner, value, note);
    }

    function hintBatchBurnWhenMoveToMainnet(address[] burners, uint256[] values, string note) onlyOwner public returns (bool ret) {
        uint256 length = burners.length;
        require(length == values.length, "The size of \'burners\' and \'values\' array is different.");

        ret = true;
        for (uint256 i = 0; i < length; i++) {
            ret = ret && hintBurnWhenMoveToMainnet(burners[i], values[i], note);
        }
    }

    /**
     * dev           HINT
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //CTK FAIL NO_ASF
    /*@CTK "hintSell correctness"
      @tag assume_completion
      @post to != 0x0
      @post to != address(this)
      @post owners[msg.sender] == true
      @post value <= balances[from] && value <= allowed[from][msg.sender]
      @post (value <= balances[from] - delayLockValues[from])
            || (value <= balances[from] - delayLockBeforeValues[from])
      @post value <= balances[from] - lockValues[from]
      @post (!locked || unlockAddrs[from])
      @post to != from -> __post.balances[from] == balances[from] - value
      @post to != from -> __post.balances[to] == balances[to] + value
      @post to == from -> __post.balances[from] == balances[from]
      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
     */
    function hintSell(
        address from,
        address to,
        uint256 value,
        string note
    ) onlyOwner public returns (bool ret) {
        require(to != address(this), "The receive address is the Contact Address of
            HINTToken. You cannot send money to this address.");

        ret = hintTransferFrom(from, to, value, note);
        emit HINTSell(from, msg.sender, to, value, note);
    }

    /**
     * dev                                        HINT
     * dev EOA
                             . (                        )
     */
    function hintBatchSellByOtherCoin(
        address from,
        address[] to,
        uint256[] values,
        uint256 processIdHash,
        uint256[] userIdHash,
        string note
    ) onlyOwner public returns (bool ret) {
        uint256 length = to.length;
        require(length == values.length, "The size of \'to\' and \'values\' array is
            different.");
        require(length == userIdHash.length, "The size of \'to\' and \'userIdHash\'
            array is different.");

        ret = true;
        for (uint256 i = 0; i < length; i++) {
            require(to[i] != address(this), "The receive address is the Contact Address
                of HINTToken. You cannot send money to this address.");

            ret = ret && hintTransferFrom(from, to[i], values[i], note);
            emit HINTSellByOtherCoin(from, msg.sender, to[i], values[i], processIdHash,
                 userIdHash[i], note);
        }
    }

    /**
     * dev
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //CTK NO_ASF
    /*@CTK "hintTransferToTeam correctness"
      @tag assume_completion
      @post to != 0x0
      @post to != address(this)
      @post owners[msg.sender] == true
      @post value <= balances[from] && value <= allowed[from][msg.sender]
      @post (value <= balances[from] - delayLockValues[from])
            || (value <= balances[from] - delayLockBeforeValues[from])
      @post value <= balances[from] - lockValues[from]
      @post (!locked || unlockAddrs[from])
      @post to != from -> __post.balances[from] == balances[from] - value
      @post to != from -> __post.balances[to] == balances[to] + value
      @post to == from -> __post.balances[from] == balances[from]
      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
     */
    function hintTransferToTeam(
        address from,
        address to,
        uint256 value,
        string note
    ) onlyOwner public returns (bool ret) {
        require(to != address(this), "The receive address is the Contact Address of
            HINTToken. You cannot send money to this address.");

        ret = hintTransferFrom(from, to, value, note);
        emit HINTTransferToTeam(from, msg.sender, to, value, note);
    }

    /**
     * dev
     */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //CTK FAIL NO_ASF
    /*@CTK "hintTransferToPartner correctness"
      @tag assume_completion
      @post to != 0x0
      @post to != address(this)
      @post owners[msg.sender] == true
      @post value <= balances[from] && value <= allowed[from][msg.sender]
      @post (value <= balances[from] - delayLockValues[from])
            || (value <= balances[from] - delayLockBeforeValues[from])
      @post value <= balances[from] - lockValues[from]
      @post (!locked || unlockAddrs[from])
      @post to != from -> __post.balances[from] == balances[from] - value
      @post to != from -> __post.balances[to] == balances[to] + value
      @post to == from -> __post.balances[from] == balances[from]
      @post __post.allowed[from][msg.sender] == allowed[from][msg.sender] - value
     */
    function hintTransferToPartner(
        address from,
        address to,
        uint256 value,
        string note
    ) onlyOwner public returns (bool ret) {
        require(to != address(this), "The receive address is the Contact Address of
            HINTToken. You cannot send money to this address.");

        ret = hintTransferFrom(from, to, value, note);
        emit HINTTransferToPartner(from, msg.sender, to, value, note);
    }

    /**
     * dev                (                                    )      HINT
     * dev EOA
                              . (                        )
     */
    function hintBatchTransferToEcosystem(
        address from, address[] to,
        uint256[] values,
        uint256 processIdHash,
        uint256[] userIdHash,
        string note
    ) onlyOwner public returns (bool ret) {
        uint256 length = to.length;
        require(length == values.length, "The size of \'to\' and \'values\' array is
            different.");
        require(length == userIdHash.length, "The size of \'to\' and \'userIdHash\'
            array is different.");

        ret = true;
        for (uint256 i = 0; i < length; i++) {
            require(to[i] != address(this), "The receive address is the Contact Address
                of HINTToken. You cannot send money to this address.");

            ret = ret && hintTransferFrom(from, to[i], values[i], note);
            emit HINTTransferToEcosystem(from, msg.sender, to[i], values[i],
                processIdHash, userIdHash[i], note);
        }
    }

    /**
     * dev                          HINT
     * dev EOA
                              . (                        )
     */
    function hintBatchTransferToBounty(
        address from,
        address[] to,
        uint256[] values,
        uint256 processIdHash,
        uint256[] userIdHash,
        string note
    ) onlyOwner public returns (bool ret) {
        uint256 length = to.length;
        require(to.length == values.length, "The size of \'to\' and \'values\' array is
             different.");

        ret = true;
        for (uint256 i = 0; i < length; i++) {
            require(to[i] != address(this), "The receive address is the Contact Address
                 of HINTToken. You cannot send money to this address.");

            ret = ret && hintTransferFrom(from, to[i], values[i], note);
            emit HINTTransferToBounty(from, msg.sender, to[i], values[i], processIdHash
                , userIdHash[i], note);
        }
    }

    function destroy() onlyRoot public {
        selfdestruct(root);
    }
}

/**
 * @title HINTToken
 */
contract HINTToken is HINTBaseToken {
    using AddressUtils for address;

    event TransferedToHINTDapp(
        address indexed owner,
        address indexed spender,
        address indexed to, uint256 value, HINTReceiver.HINTReceiveType receiveType);

    string public constant name = "HINT Token";
    string public constant symbol = "HINT";
    uint8 public constant decimals = 18;

    uint256 public constant INITIAL_SUPPLY = 1e9 * (10 ** uint256(decimals));

    /*@CTK "constructor correctness"
      @tag assume_completion
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post __post.totalSupply_ == INITIAL_SUPPLY
      @post __post.balances[msg.sender] == __post.totalSupply_
     */
    constructor() public {
        totalSupply_ = INITIAL_SUPPLY;
        balances[msg.sender] = INITIAL_SUPPLY;
        emit Transfer(0x0, msg.sender, INITIAL_SUPPLY);
    }

    function hintTransfer(address to, uint256 value, string note) public returns (bool
         ret) {
        ret = super.hintTransfer(to, value, note);
        postTransfer(msg.sender, msg.sender, to, value, HINTReceiver.HINTReceiveType.
            HINT_TRANSFER);
    }

    function hintTransferFrom(address from, address to, uint256 value, string note)
        public returns (bool ret) {
        ret = super.hintTransferFrom(from, to, value, note);
        postTransfer(from, msg.sender, to, value, HINTReceiver.HINTReceiveType.
            HINT_TRANSFER);
    }

    function postTransfer(address owner, address spender, address to, uint256 value,
        HINTReceiver.HINTReceiveType receiveType) internal returns (bool) {
        if (to.isContract()) {
            bool callOk = address(to).call(bytes4(keccak256("onHINTReceived(address,
                address,uint256,uint8)")), owner, spender, value, receiveType);
            if (callOk) {
                emit TransferedToHINTDapp(owner, spender, to, value, receiveType);
            }
        }

        return true;
    }

    function hintMintTo(address to, uint256 amount, string note) onlyOwner public
        returns (bool ret) {
        ret = super.hintMintTo(to, amount, note);
        postTransfer(0x0, msg.sender, to, amount, HINTReceiver.HINTReceiveType.
            HINT_MINT);
    }

    function hintBurnFrom(address from, uint256 value, string note) onlyOwner public
        returns (bool ret) {
        ret = super.hintBurnFrom(from, value, note);
        postTransfer(0x0, msg.sender, from, value, HINTReceiver.HINTReceiveType.
            HINT_BURN);
    }
}


/**
 * @title HINTToken Receiver
 */
contract HINTReceiver {
    enum HINTReceiveType { HINT_TRANSFER, HINT_MINT, HINT_BURN }
    function onHINTReceived(address owner, address spender, uint256 value,
        HINTReceiveType receiveType) public returns (bool);
}

/**
 * @title HINTDappSample
 */
contract HINTDappSample is HINTReceiver {
    event LogOnReceiveHINT(string message, address indexed owner, address indexed
        spender, uint256 value, HINTReceiveType receiveType);

    /*@CTK "onHINTReceived correctness"
      @post !(__has_overflow)
      @post !(__has_buf_overflow)
      @post !(__has_assertion_failure)
      @post __return == true
```

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