CERTIK

Audit Report

Produced by CertiK

for Vidy

Aug 29th, 2019

Contents

Disclaimer

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

About CertiK

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

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

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

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

Exective Summary

This report has been prepared as the product of the Smart Contract Audit request by Vidy. This audit was conducted to discover issues and vulnerabilities in the source code of Vidy'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 issue 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 conditions, or it could affect the security standard by lost of access control.

Low

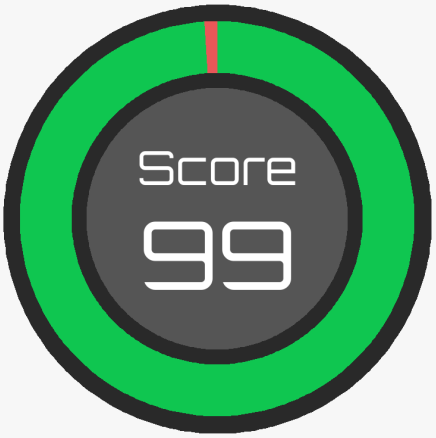
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerabilies, but no concern found yet.

Testing Summary

PASS

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

Aug 29, 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.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	1	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

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

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found.

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

`VidyCoin` Smart Contract[1]:

- **VidyCoin.sol**
  `16ed54f763b0cf558b099e3a7e93629d199a0d8e75af182122dd515e8079dc71`

`VidyListingEscrow` Smart Contract[2]:

- **Address.sol**
  `61a3b17ecec60de6d5e07c89e7017bd3b400649a42f42f5f12c3117d07af5b68`
- **IERC20.sol**
  `23221a896472eeee23d71500d71f40bcce31112b9198389310d2e7ff7d0be093`
- **SafeERC20.sol**
  `f77c89a78b55c3d6386a055afc3e753e56aa79860992a30c0a4290d48136b996`
- **SafeMath.sol**
  `469b57d4f3c4e1d39e117ea6839a987e2a6e5b2fde6cce7a72e609df8b7b1443`
- **TokenTimelock.sol**
  `41f4d330ad9d831862ec5a4c2b8a3938ce698e61daa825a17eb39e255f1d6068`
- **VidyListingEscrow.sol**
  `c1f2594cee65ff770938bb77fc8ad8914c2de9d076538ee42e683965709febb6`

Summary

CertiK was chosen by Vidy to audit the design and implementation of its `VidyCoin` and `VidyListingEscrow` smart contract. To ensure comprehensive protection,the source code has been analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space.

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

[1]Etherscan: `https://etherscan.io/address/0x79ca4a5285e477f5ccec2361fdfc13c038810ecb`
[2]Etherscan: `https://etherscan.io/address/0xa721920bf3c9d1de8d3aa7c89bb7081a61034ac8`

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File TokenTimelock.sol

```
1 pragma solidity ^0.5.0;
```

Only these compiler versions are safe to compile your code: 0.5.10

TIMESTAMP_DEPENDENCY

Line 73 in File TokenTimelock.sol

```
73        require(block.timestamp >= _releaseTime, "TokenTimelock: current time is before
              release time");
```

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

INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

```
1 pragma solidity ^0.5.0;
```

Only these compiler versions are safe to compile your code: 0.5.10

INSECURE_COMPILER_VERSION

Line 5 in File VidyCoin.sol

```
5 pragma solidity ^0.4.23;
```

Version to compile has the following bug: 0.4.23: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement, DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.24: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement, DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: SignedArrayStorageCopy, ABIEncoderV2StorageArrayW DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: SignedArrayStorageCopy, ABIEncoderV2StorageArrayWithMultiSlotElement, DynamicConstructorArgumentsClippedABIV2

Formal Verification Results

How to read

Detail for Request 1

transferFrom to same address

Verification date	20, Oct 2018
Verification timespan	395.38 ms

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

CERTIK *label*

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

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

Raw code

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

Counterexample	This code violates the specification

Initial environment

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

Post environment

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

Formal Verification Request 1

If method completes, integer overflow would not happen.

29, Aug 2019
17.03 ms

Line 22 in File TokenTimelock.sol

```
22    //@CTK NO_OVERFLOW
```

Line 32-38 in File TokenTimelock.sol

```
32    constructor (IERC20 token, address beneficiary, uint256 releaseTime) public {
33        // solhint-disable-next-line not-rely-on-time
34        require(releaseTime > block.timestamp, "TokenTimelock: release time is before
              current time");
35        _token = token;
36        _beneficiary = beneficiary;
37        _releaseTime = releaseTime;
38    }
```

The code meets the specification.

Formal Verification Request 2

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

29, Aug 2019
0.37 ms

Line 23 in File TokenTimelock.sol

```
23    //@CTK NO_BUF_OVERFLOW
```

Line 32-38 in File TokenTimelock.sol

```
32    constructor (IERC20 token, address beneficiary, uint256 releaseTime) public {
33        // solhint-disable-next-line not-rely-on-time
34        require(releaseTime > block.timestamp, "TokenTimelock: release time is before
              current time");
35        _token = token;
36        _beneficiary = beneficiary;
37        _releaseTime = releaseTime;
38    }
```

The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.

29, Aug 2019
0.35 ms

Line 24 in File TokenTimelock.sol

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

Line 32-38 in File TokenTimelock.sol

```
32    constructor (IERC20 token, address beneficiary, uint256 releaseTime) public {
33        // solhint-disable-next-line not-rely-on-time
34        require(releaseTime > block.timestamp, "TokenTimelock: release time is before
              current time");
35        _token = token;
36        _beneficiary = beneficiary;
37        _releaseTime = releaseTime;
38    }
```

The code meets the specification.

Formal Verification Request 4

TokenTimelock

29, Aug 2019
2.22 ms

Line 25-31 in File TokenTimelock.sol

```
25    /*@CTK TokenTimelock
26      @tag assume_completion
27      @post releaseTime > block.timestamp
28      @post __post._token == token
29      @post __post._beneficiary == beneficiary
30      @post __post._releaseTime == releaseTime
31     */
```

Line 32-38 in File TokenTimelock.sol

```
32    constructor (IERC20 token, address beneficiary, uint256 releaseTime) public {
33        // solhint-disable-next-line not-rely-on-time
34        require(releaseTime > block.timestamp, "TokenTimelock: release time is before
              current time");
35        _token = token;
36        _beneficiary = beneficiary;
37        _releaseTime = releaseTime;
38    }
```

The code meets the specification.

Formal Verification Request 5

SafeMath add

29, Aug 2019
14.27 ms

Line 26-34 in File SafeMath.sol

```
26    /*@CTK "SafeMath add"
27      @tag spec
28      @tag is_pure
```

```
29      @post (a + b < a || a + b < b) == __reverted
30      @post !__reverted -> __return == a + b
31      @post !__reverted -> !__has_overflow
32      @post !__reverted -> !__has_assertion_failure
33      @post !(__has_buf_overflow)
34    */
```

Line 35-40 in File SafeMath.sol

```
35    function add(uint256 a, uint256 b) internal pure returns (uint256) {
36        uint256 c = a + b;
37        require(c >= a, "SafeMath: addition overflow");
38
39        return c;
40    }
```

The code meets the specification.

Formal Verification Request 6

SafeMath sub

29, Aug 2019
13.52 ms

Line 51-59 in File SafeMath.sol

```
51    /*@CTK "SafeMath sub"
52      @tag spec
53      @tag is_pure
54      @post (b > a) == __reverted
55      @post !__reverted -> __return == a - b
56      @post !__reverted -> !__has_overflow
57      @post !__reverted -> !__has_assertion_failure
58      @post !(__has_buf_overflow)
59    */
```

Line 60-65 in File SafeMath.sol

```
60    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
61        require(b <= a, "SafeMath: subtraction overflow");
62        uint256 c = a - b;
63
64        return c;
65    }
```

The code meets the specification.

Formal Verification Request 7

SafeMath mul zero

29, Aug 2019
16.63 ms

Line 76-81 in File SafeMath.sol

```
/*@CTK "SafeMath mul zero"
  @tag spec
  @tag is_pure
  @pre (a == 0)
  @post __return == 0
*/
```

Line 92-104 in File SafeMath.sol

```
function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    // Gas optimization: this is cheaper than requiring '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;
    }

    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");

    return c;
}
```

The code meets the specification.

Formal Verification Request 8

SafeMath mul nonzero

29, Aug 2019
285.22 ms

Line 82-91 in File SafeMath.sol

```
/*@CTK "SafeMath mul nonzero"
  @tag spec
  @tag is_pure
  @pre (a != 0)
  @post (a * b / a != b) == __reverted
  @post !__reverted -> __return == a * b
  @post !__reverted -> !__has_overflow
  @post !__reverted -> !__has_assertion_failure
  @post !(__has_buf_overflow)
*/
```

Line 92-104 in File SafeMath.sol

```
function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    // Gas optimization: this is cheaper than requiring '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;
    }

    uint256 c = a * b;
    require(c / a == b, "SafeMath: multiplication overflow");

```

```
        return c;
    }
```

The code meets the specification.

Formal Verification Request 9

SafeMath div

29, Aug 2019
14.77 ms

Line 117-125 in File SafeMath.sol

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

Line 126-133 in File SafeMath.sol

```
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0, "SafeMath: division by zero");
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }
```

The code meets the specification.

Formal Verification Request 10

SafeMath mod

29, Aug 2019
12.47 ms

Line 146-154 in File SafeMath.sol

```
    /*@CTK "SafeMath mod"
      @tag spec
      @tag is_pure
      @post (b == 0) == __reverted
      @post !__reverted -> __return == a % b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
```

Line 155-158 in File SafeMath.sol

```
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0, "SafeMath: modulo by zero");
        return a % b;
    }
```

The code meets the specification.

Formal Verification Request 11

Method will not encounter an assertion failure.

29, Aug 2019
19.52 ms

Line 12 in File VidyCoin.sol

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

Line 20-31 in File VidyCoin.sol

```
20  function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
21    // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
22    // benefit is lost if 'b' is also tested.
23    // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
24    if (a == 0) {
25      return 0;
26    }
27
28    c = a * b;
29    assert(c / a == b);
30    return c;
31  }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          a = 2
5          b = 156
6      }
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14           "gas": 0,
15           "sender": 0,
16           "value": 0
17         }
18     }
19     Other = {
20         block = {
21           "number": 0,
22           "timestamp": 0
23         }
```

```
        c = 0
    }
    Address_Map = [
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 12

SafeMath mul

29, Aug 2019
273.06 ms

Line 13-19 in File VidyCoin.sol

```
13  /*@CTK "SafeMath mul"
14    @post ((a > 0) && (((a * b) / a) != b)) == (__reverted)
15    @post !__reverted -> c == a * b
16    @post !__reverted == !__has_overflow
17    @post !__reverted -> !(__has_assertion_failure)
18    @post !(__has_buf_overflow)
19  */
```

Line 20-31 in File VidyCoin.sol

```
20  function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
21    // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
22    // benefit is lost if 'b' is also tested.
23    // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
24    if (a == 0) {
25      return 0;
26    }
27
28    c = a * b;
29    assert(c / a == b);
30    return c;
31  }
```

The code meets the specification.

Formal Verification Request 13

Method will not encounter an assertion failure.

29, Aug 2019
5.5 ms

Line 36 in File VidyCoin.sol

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

Line 44-49 in File VidyCoin.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;
}
```

This code violates the specification.

```
Counter Example:
Before Execution:
   Input = {
       a = 0
       b = 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": 0
       }
   }
   Address_Map = [
     {
       "key": "ALL_OTHERS",
       "value": "EmptyAddress"
     }
   ]

Function invocation is reverted.
```

Formal Verification Request 14

SafeMath div

29, Aug 2019
0.32 ms

Line 37-43 in File VidyCoin.sol

```
37 /*@CTK "SafeMath div"
38   @post b != 0 -> !__reverted
39   @post !__reverted -> __return == a / b
40   @post !__reverted -> !__has_overflow
41   @post !__reverted -> !(__has_assertion_failure)
42   @post !(__has_buf_overflow)
```

```
43  */
```

Line 44-49 in File VidyCoin.sol

```
44  function div(uint256 a, uint256 b) internal pure returns (uint256) {
45    // assert(b > 0); // Solidity automatically throws when dividing by 0
46    // uint256 c = a / b;
47    // assert(a == b * c + a % b); // There is no case in which this doesn't hold
48    return a / b;
49  }
```

✅ The code meets the specification.

Formal Verification Request 15

Method will not encounter an assertion failure.

📅 29, Aug 2019
⏱ 11.23 ms

Line 54 in File VidyCoin.sol

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

Line 62-65 in File VidyCoin.sol

```
62  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
63    assert(b <= a);
64    return a - b;
65  }
```

❌ This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          a = 0
5          b = 1
6      }
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14           "gas": 0,
15           "sender": 0,
16           "value": 0
17         }
18     }
19     Other = {
20         __return = 0
21         block = {
22           "number": 0,
23           "timestamp": 0
24         }
25     }
26     Address_Map = [
```

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

Function invocation is reverted.
```

Formal Verification Request 16

SafeMath sub

29, Aug 2019
0.81 ms

Line 55-61 in File VidyCoin.sol

```
55  /*@CTK "SafeMath sub"
56    @post (a < b) == __reverted
57    @post !__reverted -> __return == a - b
58    @post !__reverted -> !__has_overflow
59    @post !__reverted -> !(__has_assertion_failure)
60    @post !(__has_buf_overflow)
61  */
```

Line 62-65 in File VidyCoin.sol

```
62  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
63    assert(b <= a);
64    return a - b;
65  }
```

The code meets the specification.

Formal Verification Request 17

Method will not encounter an assertion failure.

29, Aug 2019
12.22 ms

Line 70 in File VidyCoin.sol

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

Line 77-81 in File VidyCoin.sol

```
77  function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
78    c = a + b;
79    assert(c >= a);
80    return c;
81  }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3     Input = {
```

```
        a = 191
        b = 65
    }
    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 18

SafeMath add

29, Aug 2019
2.72 ms

Line 71-76 in File VidyCoin.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 77-81 in File VidyCoin.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 19

renounceOwnership

29, Aug 2019
11.52 ms

Line 145-149 in File VidyCoin.sol

```
145 /*@CTK renounceOwnership
146   @tag assume_completion
147   @post msg.sender == owner
148   @post __post.owner == address(0)
149  */
```

Line 150-153 in File VidyCoin.sol

```
150 function renounceOwnership() public onlyOwner {
151   emit OwnershipRenounced(owner);
152   owner = address(0);
153 }
```

The code meets the specification.

Formal Verification Request 20

transferOwnership

29, Aug 2019
11.11 ms

Line 167-171 in File VidyCoin.sol

```
167 /*@CTK transferOwnership
168   @tag assume_completion
169   @post _newOwner != address(0)
170   @post __post.owner == _newOwner
171  */
```

Line 172-176 in File VidyCoin.sol

```
172 function _transferOwnership(address _newOwner) internal {
173   require(_newOwner != address(0));
174   emit OwnershipTransferred(owner, _newOwner);
175   owner = _newOwner;
176 }
```

The code meets the specification.

Formal Verification Request 21

If method completes, integer overflow would not happen.

29, Aug 2019
4.3 ms

Line 193 in File VidyCoin.sol

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

Line 200-202 in File VidyCoin.sol

```
200 function totalSupply() public view returns (uint256) {
201   return totalSupply_;
202 }
```

✅ The code meets the specification.

Formal Verification Request 22

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

📅 29, Aug 2019
⏱ 0.29 ms

Line 194 in File VidyCoin.sol

```
194 //@CTK NO_BUF_OVERFLOW
```

Line 200-202 in File VidyCoin.sol

```
200 function totalSupply() public view returns (uint256) {
201   return totalSupply_;
202 }
```

✅ The code meets the specification.

Formal Verification Request 23

Method will not encounter an assertion failure.

📅 29, Aug 2019
⏱ 0.28 ms

Line 195 in File VidyCoin.sol

```
195 //@CTK NO_ASF
```

Line 200-202 in File VidyCoin.sol

```
200 function totalSupply() public view returns (uint256) {
201   return totalSupply_;
202 }
```

✅ The code meets the specification.

Formal Verification Request 24

totalSupply

📅 29, Aug 2019
⏱ 0.29 ms

Line 196-199 in File VidyCoin.sol

```
196 /*@CTK totalSupply
197   @tag assume_completion
198   @post (__return) == (totalSupply_)
199 */
```

Line 200-202 in File VidyCoin.sol

```
200 function totalSupply() public view returns (uint256) {
201   return totalSupply_;
202 }
```

✅ The code meets the specification.

Formal Verification Request 25

If method completes, integer overflow would not happen.

📅 29, Aug 2019
⏱ 76.44 ms

Line 209 in File VidyCoin.sol

```
209 //@CTK NO_OVERFLOW
```

Line 222-230 in File VidyCoin.sol

```
222 function transfer(address _to, uint256 _value) public returns (bool) {
223   require(_to != address(0));
224   require(_value <= balances[msg.sender]);
225
226   balances[msg.sender] = balances[msg.sender].sub(_value);
227   balances[_to] = balances[_to].add(_value);
228   emit Transfer(msg.sender, _to, _value);
229   return true;
230 }
```

✅ The code meets the specification.

Formal Verification Request 26

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

📅 29, Aug 2019
⏱ 15.71 ms

Line 210 in File VidyCoin.sol

```
210 //@CTK NO_BUF_OVERFLOW
```

Line 222-230 in File VidyCoin.sol

```
222 function transfer(address _to, uint256 _value) public returns (bool) {
223   require(_to != address(0));
224   require(_value <= balances[msg.sender]);
225
226   balances[msg.sender] = balances[msg.sender].sub(_value);
227   balances[_to] = balances[_to].add(_value);
228   emit Transfer(msg.sender, _to, _value);
```

```
    return true;
  }
```

The code meets the specification.

Formal Verification Request 27

Method will not encounter an assertion failure.

29, Aug 2019
38.33 ms

Line 211 in File VidyCoin.sol

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

Line 222-230 in File VidyCoin.sol

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

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

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        _to = 4
        _value = 136
    }
    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": 0
        }
    }
    Address_Map = [
      {
```

```
      "key": 0,
      "value": {
        "contract_name": "BasicToken",
        "balance": 0,
        "contract": {
          "balances": [
            {
              "key": 0,
              "value": 144
            },
            {
              "key": 4,
              "value": 178
            },
            {
              "key": "ALL_OTHERS",
              "value": 8
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```


Formal Verification Request 28

transfer

29, Aug 2019
117.83 ms

Line 212-221 in File VidyCoin.sol

```
/*@CTK transfer
  @tag assume_completion
  @pre _to != address(0)
  @pre _value <= balances[msg.sender]
  @post (msg.sender != _to) -> (__post.balances[_to] == balances[_to] + _value)
  @post (msg.sender != _to) -> (__post.balances[msg.sender] == balances[msg.sender]
      - _value)
  @post (msg.sender == _to) -> (__post.balances[_to] == balances[_to])
  @post (msg.sender == _to) -> (__post.balances[msg.sender] == balances[msg.sender])
  @post __return == true
*/
```

Line 222-230 in File VidyCoin.sol

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

```

```
226     balances[msg.sender] = balances[msg.sender].sub(_value);
227     balances[_to] = balances[_to].add(_value);
228     emit Transfer(msg.sender, _to, _value);
229     return true;
230   }
```

The code meets the specification.

Formal Verification Request 29

If method completes, integer overflow would not happen.

29, Aug 2019
4.99 ms

Line 237 in File VidyCoin.sol

```
237   //@CTK NO_OVERFLOW
```

Line 244-246 in File VidyCoin.sol

```
244   function balanceOf(address _owner) public view returns (uint256) {
245     return balances[_owner];
246   }
```

The code meets the specification.

Formal Verification Request 30

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

29, Aug 2019
0.36 ms

Line 238 in File VidyCoin.sol

```
238   //@CTK NO_BUF_OVERFLOW
```

Line 244-246 in File VidyCoin.sol

```
244   function balanceOf(address _owner) public view returns (uint256) {
245     return balances[_owner];
246   }
```

The code meets the specification.

Formal Verification Request 31

Method will not encounter an assertion failure.

29, Aug 2019
0.3 ms

Line 239 in File VidyCoin.sol

```
239   //@CTK NO_ASF
```

Line 244-246 in File VidyCoin.sol

```
244 function balanceOf(address _owner) public view returns (uint256) {
245   return balances[_owner];
246 }
```

The code meets the specification.

Formal Verification Request 32

balanceOf

29, Aug 2019
0.32 ms

Line 240-243 in File VidyCoin.sol

```
240 /*@CTK balanceOf
241   @tag assume_completion
242   @post (__return) == (balances[_owner])
243 */
```

Line 244-246 in File VidyCoin.sol

```
244 function balanceOf(address _owner) public view returns (uint256) {
245   return balances[_owner];
246 }
```

The code meets the specification.

Formal Verification Request 33

If method completes, integer overflow would not happen.

29, Aug 2019
111.58 ms

Line 263 in File VidyCoin.sol

```
263 //@CTK NO_OVERFLOW
```

Line 272-274 in File VidyCoin.sol

```
272 function burn(uint256 _value) public {
273   _burn(msg.sender, _value);
274 }
```

The code meets the specification.

Formal Verification Request 34

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

29, Aug 2019
8.5 ms

Line 264 in File VidyCoin.sol

```
264 //@CTK NO_BUF_OVERFLOW
```

Line 272-274 in File VidyCoin.sol

```
272 function burn(uint256 _value) public {
273   _burn(msg.sender, _value);
274 }
```

✅ The code meets the specification.

Formal Verification Request 35

Method will not encounter an assertion failure.

📅 29, Aug 2019
⏱ 18.31 ms

Line 265 in File VidyCoin.sol

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

Line 272-274 in File VidyCoin.sol

```
272 function burn(uint256 _value) public {
273   _burn(msg.sender, _value);
274 }
```

❌ This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          _value = 64
5      }
6      This = 0
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14           "gas": 0,
15           "sender": 0,
16           "value": 0
17         }
18     }
19     Other = {
20         block = {
21           "number": 0,
22           "timestamp": 0
23         }
24     }
25     Address_Map = [
26       {
27         "key": 0,
28         "value": {
29           "contract_name": "BurnableToken",
30           "balance": 0,
```

```
        "contract": {
          "balances": [
            {
              "key": 0,
              "value": 65
            },
            {
              "key": "ALL_OTHERS",
              "value": 1
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 36

burn

📅 29, Aug 2019
⏱ 32.12 ms

Line 266-271 in File VidyCoin.sol

```
/*@CTK burn
  @tag assume_completion
  @post (_value <= balances[msg.sender])
  @post (__post.totalSupply_) == ((totalSupply_) - (_value))
  @post (__post.balances[msg.sender]) == ((balances[msg.sender]) - (_value))
 */
```

Line 272-274 in File VidyCoin.sol

```
function burn(uint256 _value) public {
  _burn(msg.sender, _value);
}
```

✅ The code meets the specification.

Formal Verification Request 37

If method completes, integer overflow would not happen.

📅 29, Aug 2019
⏱ 8.19 ms

Line 276 in File VidyCoin.sol

```
276 //@CTK NO_OVERFLOW
```

Line 285-294 in File VidyCoin.sol

```
285 function _burn(address _who, uint256 _value) internal {
286   require(_value <= balances[_who]);
287   // no need to require value <= totalSupply, since that would imply the
288   // sender's balance is greater than the totalSupply, which *should* be an
        assertion failure
289
290   balances[_who] = balances[_who].sub(_value);
291   totalSupply_ = totalSupply_.sub(_value);
292   emit Burn(_who, _value);
293   emit Transfer(_who, address(0), _value);
294 }
```

The code meets the specification.

Formal Verification Request 38

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

29, Aug 2019
9.88 ms

Line 277 in File VidyCoin.sol

```
277 //@CTK NO_BUF_OVERFLOW
```

Line 285-294 in File VidyCoin.sol

```
285 function _burn(address _who, uint256 _value) internal {
286   require(_value <= balances[_who]);
287   // no need to require value <= totalSupply, since that would imply the
288   // sender's balance is greater than the totalSupply, which *should* be an
        assertion failure
289
290   balances[_who] = balances[_who].sub(_value);
291   totalSupply_ = totalSupply_.sub(_value);
292   emit Burn(_who, _value);
293   emit Transfer(_who, address(0), _value);
294 }
```

The code meets the specification.

Formal Verification Request 39

Method will not encounter an assertion failure.

29, Aug 2019
20.44 ms

Line 278 in File VidyCoin.sol

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

Line 285-294 in File VidyCoin.sol

```
  function _burn(address _who, uint256 _value) internal {
    require(_value <= balances[_who]);
    // no need to require value <= totalSupply, since that would imply the
    // sender's balance is greater than the totalSupply, which *should* be an
        assertion failure

    balances[_who] = balances[_who].sub(_value);
    totalSupply_ = totalSupply_.sub(_value);
    emit Burn(_who, _value);
    emit Transfer(_who, address(0), _value);
  }
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        _value = 192
        _who = 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 = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "BurnableToken",
          "balance": 0,
          "contract": {
            "balances": [
              {
                "key": 0,
                "value": 196
              },
              {
                "key": "ALL_OTHERS",
                "value": 64
              }
            ],
            "totalSupply_": 0
          }
        }
```

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

Function invocation is reverted.
```

Formal Verification Request 40

_burn

29, Aug 2019
35.46 ms

Line 279-284 in File VidyCoin.sol

```
/*@CTK _burn
  @tag assume_completion
  @post (_value <= balances[_who])
  @post (__post.totalSupply_) == ((totalSupply_) - (_value))
  @post (__post.balances[_who]) == ((balances[_who]) - (_value))
*/
```

Line 285-294 in File VidyCoin.sol

```
function _burn(address _who, uint256 _value) internal {
  require(_value <= balances[_who]);
  // no need to require value <= totalSupply, since that would imply the
  // sender's balance is greater than the totalSupply, which *should* be an
      assertion failure

  balances[_who] = balances[_who].sub(_value);
  totalSupply_ = totalSupply_.sub(_value);
  emit Burn(_who, _value);
  emit Transfer(_who, address(0), _value);
}
```

The code meets the specification.

Formal Verification Request 41

If method completes, integer overflow would not happen.

29, Aug 2019
95.81 ms

Line 321 in File VidyCoin.sol

```
//@CTK NO_OVERFLOW
```

Line 333-350 in File VidyCoin.sol

```
function transferFrom(
  address _from,
  address _to,
```

```
336     uint256 _value
337   )
338     public
339     returns (bool)
340   {
341     require(_to != address(0));
342     require(_value <= balances[_from]);
343     require(_value <= allowed[_from][msg.sender]);
344
345     balances[_from] = balances[_from].sub(_value);
346     balances[_to] = balances[_to].add(_value);
347     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
348     emit Transfer(_from, _to, _value);
349     return true;
350   }
```

The code meets the specification.

Formal Verification Request 42

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

29, Aug 2019
13.71 ms

Line 322 in File VidyCoin.sol

```
322   //@CTK NO_BUF_OVERFLOW
```

Line 333-350 in File VidyCoin.sol

```
333   function transferFrom(
334     address _from,
335     address _to,
336     uint256 _value
337   )
338     public
339     returns (bool)
340   {
341     require(_to != address(0));
342     require(_value <= balances[_from]);
343     require(_value <= allowed[_from][msg.sender]);
344
345     balances[_from] = balances[_from].sub(_value);
346     balances[_to] = balances[_to].add(_value);
347     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
348     emit Transfer(_from, _to, _value);
349     return true;
350   }
```

The code meets the specification.

Formal Verification Request 43

Method will not encounter an assertion failure.

29, Aug 2019

135.47 ms

Line 323 in File VidyCoin.sol

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

Line 333-350 in File VidyCoin.sol

```
333 function transferFrom(
334   address _from,
335   address _to,
336   uint256 _value
337 )
338   public
339   returns (bool)
340 {
341   require(_to != address(0));
342   require(_value <= balances[_from]);
343   require(_value <= allowed[_from][msg.sender]);
344
345   balances[_from] = balances[_from].sub(_value);
346   balances[_to] = balances[_to].add(_value);
347   allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
348   emit Transfer(_from, _to, _value);
349   return true;
350 }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3      Input = {
4          _from = 0
5          _to = 2
6          _value = 17
7      }
8      This = 0
9      Internal = {
10         __has_assertion_failure = false
11         __has_buf_overflow = false
12         __has_overflow = false
13         __has_returned = false
14         __reverted = false
15         msg = {
16           "gas": 0,
17           "sender": 0,
18           "value": 0
19         }
20     }
21     Other = {
22         __return = false
23         block = {
24           "number": 0,
25           "timestamp": 0
26         }
27     }
28     Address_Map = [
29       {
30         "key": 0,
31         "value": {
```

```
        "contract_name": "StandardToken",
        "balance": 0,
        "contract": {
          "allowed": [
            {
              "key": 0,
              "value": [
                {
                  "key": 0,
                  "value": 128
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 17
                }
              ]
            },
            {
              "key": 64,
              "value": [
                {
                  "key": 0,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            },
            {
              "key": 8,
              "value": [
                {
                  "key": 0,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 128
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 17
                }
              ]
            }
          ],
          "balances": [
            {
              "key": 1,
              "value": 0
            },
```

```
            {
              "key": 0,
              "value": 128
            },
            {
              "key": 2,
              "value": 241
            },
            {
              "key": 16,
              "value": 0
            },
            {
              "key": "ALL_OTHERS",
              "value": 17
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 44

transferFrom correctness

📅 29, Aug 2019
⏱ 372.41 ms

Line 324-332 in File VidyCoin.sol

```
/*@CTK "transferFrom correctness"
  @tag assume_completion
  @post _to != 0x0
  @post _value <= balances[_from] && _value <= allowed[_from][msg.sender]
  @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 333-350 in File VidyCoin.sol

```
function transferFrom(
  address _from,
  address _to,
  uint256 _value
)
  public
  returns (bool)
{
```

```
    require(_to != address(0));
    require(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

    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;
  }
```

The code meets the specification.

Formal Verification Request 45

If method completes, integer overflow would not happen.

29, Aug 2019
8.66 ms

Line 362 in File VidyCoin.sol

```
362 //@CTK NO_OVERFLOW
```

Line 368-372 in File VidyCoin.sol

```
368 function approve(address _spender, uint256 _value) public returns (bool) {
369   allowed[msg.sender][_spender] = _value;
370   emit Approval(msg.sender, _spender, _value);
371   return true;
372 }
```

The code meets the specification.

Formal Verification Request 46

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

29, Aug 2019
0.38 ms

Line 363 in File VidyCoin.sol

```
363 //@CTK NO_BUF_OVERFLOW
```

Line 368-372 in File VidyCoin.sol

```
368 function approve(address _spender, uint256 _value) public returns (bool) {
369   allowed[msg.sender][_spender] = _value;
370   emit Approval(msg.sender, _spender, _value);
371   return true;
372 }
```

The code meets the specification.

Formal Verification Request 47

Method will not encounter an assertion failure.

29, Aug 2019
0.35 ms

Line 364 in File VidyCoin.sol

```
364 //@CTK NO_ASF
```

Line 368-372 in File VidyCoin.sol

```
368 function approve(address _spender, uint256 _value) public returns (bool) {
369   allowed[msg.sender][_spender] = _value;
370   emit Approval(msg.sender, _spender, _value);
371   return true;
372 }
```

The code meets the specification.

Formal Verification Request 48

approve correctness

29, Aug 2019
1.16 ms

Line 365-367 in File VidyCoin.sol

```
365 /*@CTK "approve correctness"
366   @post __post.allowed[msg.sender][_spender] == _value
367  */
```

Line 368-372 in File VidyCoin.sol

```
368 function approve(address _spender, uint256 _value) public returns (bool) {
369   allowed[msg.sender][_spender] = _value;
370   emit Approval(msg.sender, _spender, _value);
371   return true;
372 }
```

The code meets the specification.

Formal Verification Request 49

If method completes, integer overflow would not happen.

29, Aug 2019
4.36 ms

Line 380 in File VidyCoin.sol

```
380 //@CTK NO_OVERFLOW
```

Line 386-395 in File VidyCoin.sol

```
386 function allowance(
387   address _owner,
388   address _spender
389  )
390   public
391   view
392   returns (uint256)
393 {
394   return allowed[_owner][_spender];
395 }
```

✅ The code meets the specification.

Formal Verification Request 50

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

📅 29, Aug 2019
⏱ 0.39 ms

Line 381 in File VidyCoin.sol

```
381 //@CTK NO_BUF_OVERFLOW
```

Line 386-395 in File VidyCoin.sol

```
386 function allowance(
387   address _owner,
388   address _spender
389  )
390   public
391   view
392   returns (uint256)
393 {
394   return allowed[_owner][_spender];
395 }
```

✅ The code meets the specification.

Formal Verification Request 51

Method will not encounter an assertion failure.

📅 29, Aug 2019
⏱ 0.4 ms

Line 382 in File VidyCoin.sol

```
382 //@CTK NO_ASF
```

Line 386-395 in File VidyCoin.sol

```
386 function allowance(
387   address _owner,
388   address _spender
389  )
390   public
```

```
    view
    returns (uint256)
  {
    return allowed[_owner][_spender];
  }
```

The code meets the specification.

Formal Verification Request 52

allowance correctness

29, Aug 2019
0.31 ms

Line 383-385 in File VidyCoin.sol

```
383  /*@CTK "allowance correctness"
384    @post __return == allowed[_owner][_spender]
385   */
```

Line 386-395 in File VidyCoin.sol

```
386  function allowance(
387    address _owner,
388    address _spender
389   )
390    public
391    view
392    returns (uint256)
393  {
394    return allowed[_owner][_spender];
395  }
```

The code meets the specification.

Formal Verification Request 53

If method completes, integer overflow would not happen.

29, Aug 2019
29.18 ms

Line 408 in File VidyCoin.sol

```
408  //@CTK NO_OVERFLOW
```

Line 415-426 in File VidyCoin.sol

```
415  function increaseApproval(
416    address _spender,
417    uint _addedValue
418  )
419    public
420    returns (bool)
421  {
422    allowed[msg.sender][_spender] = (
```

```
    allowed[msg.sender][_spender].add(_addedValue));
  emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
  return true;
}
```

The code meets the specification.

Formal Verification Request 54

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

29, Aug 2019
0.59 ms

Line 409 in File VidyCoin.sol

```
409 //@CTK NO_BUF_OVERFLOW
```

Line 415-426 in File VidyCoin.sol

```
415 function increaseApproval(
416   address _spender,
417   uint _addedValue
418 )
419   public
420   returns (bool)
421 {
422   allowed[msg.sender][_spender] = (
423     allowed[msg.sender][_spender].add(_addedValue));
424   emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
425   return true;
426 }
```

The code meets the specification.

Formal Verification Request 55

Method will not encounter an assertion failure.

29, Aug 2019
5.52 ms

Line 410 in File VidyCoin.sol

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

Line 415-426 in File VidyCoin.sol

```
415 function increaseApproval(
416   address _spender,
417   uint _addedValue
418 )
419   public
420   returns (bool)
421 {
422   allowed[msg.sender][_spender] = (
423     allowed[msg.sender][_spender].add(_addedValue));
```

```
    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    return true;
  }
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        _addedValue = 1
        _spender = 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": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "StandardToken",
          "balance": 0,
          "contract": {
            "allowed": [
              {
                "key": 0,
                "value": [
                  {
                    "key": 0,
                    "value": 255
                  },
                  {
                    "key": 16,
                    "value": 0
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 1
                  }
                ]
              },
              {
                "key": "ALL_OTHERS",
```

```
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 1
                }
              ]
            }
          ],
          "balances": [
            {
              "key": 16,
              "value": 2
            },
            {
              "key": "ALL_OTHERS",
              "value": 1
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 56

increaseApproval correctness

29, Aug 2019
1.69 ms

Line 411-414 in File VidyCoin.sol

```
/*@CTK "increaseApproval correctness"
  @tag assume_completion
  @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
      _addedValue
 */
```

Line 415-426 in File VidyCoin.sol

```
function increaseApproval(
  address _spender,
  uint _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;
```

```
}
```

The code meets the specification.

Formal Verification Request 57

If method completes, integer overflow would not happen.

29, Aug 2019
44.13 ms

Line 438 in File VidyCoin.sol

```
//@CTK NO_OVERFLOW
```

Line 452-467 in File VidyCoin.sol

```
function decreaseApproval(
  address _spender,
  uint _subtractedValue
)
  public
  returns (bool)
{
  uint 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;
}
```

The code meets the specification.

Formal Verification Request 58

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

29, Aug 2019
0.75 ms

Line 439 in File VidyCoin.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 452-467 in File VidyCoin.sol

```
function decreaseApproval(
  address _spender,
  uint _subtractedValue
)
  public
  returns (bool)
{
  uint oldValue = allowed[msg.sender][_spender];
```

```
460     if (_subtractedValue > oldValue) {
461       allowed[msg.sender][_spender] = 0;
462     } else {
463       allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
464     }
465     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
466     return true;
467   }
```

The code meets the specification.

Formal Verification Request 59

Method will not encounter an assertion failure.

29, Aug 2019
1.25 ms

Line 440 in File VidyCoin.sol

```
440   //@CTK NO_ASF
```

Line 452-467 in File VidyCoin.sol

```
452   function decreaseApproval(
453     address _spender,
454     uint _subtractedValue
455   )
456     public
457     returns (bool)
458   {
459     uint oldValue = allowed[msg.sender][_spender];
460     if (_subtractedValue > oldValue) {
461       allowed[msg.sender][_spender] = 0;
462     } else {
463       allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
464     }
465     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
466     return true;
467   }
```

The code meets the specification.

Formal Verification Request 60

decreaseApproval0

29, Aug 2019
21.36 ms

Line 441-445 in File VidyCoin.sol

```
441   /*@CTK decreaseApproval0
442     @pre __return == true
443     @pre allowed[msg.sender][_spender] <= _subtractedValue
444     @post __post.allowed[msg.sender][_spender] == 0
445    */
```

Line 452-467 in File VidyCoin.sol

```
452 function decreaseApproval(
453   address _spender,
454   uint _subtractedValue
455 )
456   public
457   returns (bool)
458 {
459   uint oldValue = allowed[msg.sender][_spender];
460   if (_subtractedValue > oldValue) {
461     allowed[msg.sender][_spender] = 0;
462   } else {
463     allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
464   }
465   emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
466   return true;
467 }
```

The code meets the specification.

Formal Verification Request 61

decreaseApproval

29, Aug 2019
14.32 ms

Line 446-451 in File VidyCoin.sol

```
446 /*@CTK decreaseApproval
447   @pre __return == true
448   @pre allowed[msg.sender][_spender] > _subtractedValue
449   @post __post.allowed[msg.sender][_spender] ==
450       allowed[msg.sender][_spender] - _subtractedValue
451  */
```

Line 452-467 in File VidyCoin.sol

```
452 function decreaseApproval(
453   address _spender,
454   uint _subtractedValue
455 )
456   public
457   returns (bool)
458 {
459   uint oldValue = allowed[msg.sender][_spender];
460   if (_subtractedValue > oldValue) {
461     allowed[msg.sender][_spender] = 0;
462   } else {
463     allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
464   }
465   emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
466   return true;
467 }
```

The code meets the specification.

Formal Verification Request 62

If method completes, integer overflow would not happen.

29, Aug 2019
152.26 ms

Line 484 in File VidyCoin.sol

```
484 //@CTK NO_OVERFLOW
```

Line 495-501 in File VidyCoin.sol

```
495 function burnFrom(address _from, uint256 _value) public {
496   require(_value <= allowed[_from][msg.sender]);
497   // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
      ,
498   // this function needs to emit an event with the updated approval.
499   allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
500   _burn(_from, _value);
501 }
```

The code meets the specification.

Formal Verification Request 63

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

29, Aug 2019
15.26 ms

Line 485 in File VidyCoin.sol

```
485 //@CTK NO_BUF_OVERFLOW
```

Line 495-501 in File VidyCoin.sol

```
495 function burnFrom(address _from, uint256 _value) public {
496   require(_value <= allowed[_from][msg.sender]);
497   // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
      ,
498   // this function needs to emit an event with the updated approval.
499   allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
500   _burn(_from, _value);
501 }
```

The code meets the specification.

Formal Verification Request 64

Method will not encounter an assertion failure.

29, Aug 2019
31.97 ms

Line 486 in File VidyCoin.sol

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

Line 495-501 in File VidyCoin.sol

```
function burnFrom(address _from, uint256 _value) public {
  require(_value <= allowed[_from][msg.sender]);
  // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
      ,
  // this function needs to emit an event with the updated approval.
  allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
  _burn(_from, _value);
}
```

This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        _from = 0
        _value = 192
    }
    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 = {
        block = {
          "number": 0,
          "timestamp": 0
        }
    }
    Address_Map = [
      {
        "key": 0,
        "value": {
          "contract_name": "StandardBurnableToken",
          "balance": 0,
          "contract": {
            "allowed": [
              {
                "key": 0,
                "value": [
                  {
                    "key": 8,
                    "value": 0
                  },
                  {
                    "key": 32,
                    "value": 0
                  },
                  {
```

```
                  "key": 0,
                  "value": 202
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 192
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 202
                }
              ]
            }
          ],
          "balances": [
            {
              "key": 8,
              "value": 16
            },
            {
              "key": 32,
              "value": 16
            },
            {
              "key": 0,
              "value": 192
            },
            {
              "key": "ALL_OTHERS",
              "value": 202
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```


Formal Verification Request 65

burnFrom

📅 29, Aug 2019
⏱ 129.03 ms

Line 487-494 in File VidyCoin.sol

```
/*@CTK burnFrom
  @tag assume_completion
  @post (_value <= allowed[_from][msg.sender])
  @post (_value <= balances[_from])
  @post (__post.allowed[_from][msg.sender]) == ((allowed[_from][msg.sender]) - (
      _value))
  @post (__post.balances[_from]) == ((balances[_from]) - (_value))
  @post __post.totalSupply_ == (totalSupply_ - _value)
 */
```

Line 495-501 in File VidyCoin.sol

```
function burnFrom(address _from, uint256 _value) public {
  require(_value <= allowed[_from][msg.sender]);
  // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
      ,
  // this function needs to emit an event with the updated approval.
  allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
  _burn(_from, _value);
}
```

The code meets the specification.

Formal Verification Request 66

DetailedERC20

29, Aug 2019
7.69 ms

Line 526-531 in File VidyCoin.sol

```
/*@CTK DetailedERC20
  @tag assume_completion
  @post __post.name == _name
  @post __post.symbol == _symbol
  @post __post.decimals == _decimals
 */
```

Line 532-536 in File VidyCoin.sol

```
constructor(string _name, string _symbol, uint8 _decimals) public {
  name = _name;
  symbol = _symbol;
  decimals = _decimals;
}
```

The code meets the specification.

Formal Verification Request 67

pause

29, Aug 2019
20.65 ms

Line 570-575 in File VidyCoin.sol

```
570 /*@CTK pause
571   @tag assume_completion
572   @post paused == false
573   @post owner == msg.sender
574   @post __post.paused == true
575  */
```

Line 576-579 in File VidyCoin.sol

```
576 function pause() onlyOwner whenNotPaused public {
577   paused = true;
578   emit Pause();
579 }
```

The code meets the specification.

Formal Verification Request 68

unpause

29, Aug 2019
20.99 ms

Line 584-589 in File VidyCoin.sol

```
584 /*@CTK unpause
585   @tag assume_completion
586   @post paused == true
587   @post owner == msg.sender
588   @post __post.paused == false
589  */
```

Line 590-593 in File VidyCoin.sol

```
590 function unpause() onlyOwner whenPaused public {
591   paused = false;
592   emit Unpause();
593 }
```

The code meets the specification.

Formal Verification Request 69

PausableToken_transfer

29, Aug 2019
136.21 ms

Line 604-607 in File VidyCoin.sol

```
604 /*@CTK PausableToken_transfer
605   @tag assume_completion
606   @post paused == false
607  */
```

Line 608-617 in File VidyCoin.sol

```
608 function transfer(
609   address _to,
610   uint256 _value
611 )
612   public
613   whenNotPaused
614   returns (bool)
615 {
616   return super.transfer(_to, _value);
617 }
```

The code meets the specification.

Formal Verification Request 70

PausableToken_transferFrom

29, Aug 2019
187.68 ms

Line 619-622 in File VidyCoin.sol

```
619 /*@CTK PausableToken_transferFrom
620   @tag assume_completion
621   @post paused == false
622  */
```

Line 623-633 in File VidyCoin.sol

```
623 function transferFrom(
624   address _from,
625   address _to,
626   uint256 _value
627 )
628   public
629   whenNotPaused
630   returns (bool)
631 {
632   return super.transferFrom(_from, _to, _value);
633 }
```

The code meets the specification.

Formal Verification Request 71

PausableToken_approve

29, Aug 2019
35.84 ms

Line 635-638 in File VidyCoin.sol

```
635 /*@CTK PausableToken_approve
636   @tag assume_completion
637   @post paused == false
638  */
```

Line 639-648 in File VidyCoin.sol

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

The code meets the specification.

Formal Verification Request 72

PausableToken_decreaseApproval

29, Aug 2019
74.56 ms

Line 650-653 in File VidyCoin.sol

```
/*@CTK PausableToken_decreaseApproval
  @tag assume_completion
  @post paused == false
 */
```

Line 654-663 in File VidyCoin.sol

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

The code meets the specification.

Formal Verification Request 73

If method completes, integer overflow would not happen.

29, Aug 2019
92.36 ms

Line 665 in File VidyCoin.sol

```
//@CTK NO_OVERFLOW
```

Line 672-681 in File VidyCoin.sol

```
672 function decreaseApproval(
673   address _spender,
674   uint _subtractedValue
675 )
676   public
677   whenNotPaused
678   returns (bool success)
679 {
680   return super.decreaseApproval(_spender, _subtractedValue);
681 }
```

The code meets the specification.

Formal Verification Request 74

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

29, Aug 2019
1.09 ms

Line 666 in File VidyCoin.sol

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

Line 672-681 in File VidyCoin.sol

```
672 function decreaseApproval(
673   address _spender,
674   uint _subtractedValue
675 )
676   public
677   whenNotPaused
678   returns (bool success)
679 {
680   return super.decreaseApproval(_spender, _subtractedValue);
681 }
```

The code meets the specification.

Formal Verification Request 75

Method will not encounter an assertion failure.

29, Aug 2019
1.72 ms

Line 667 in File VidyCoin.sol

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

Line 672-681 in File VidyCoin.sol

```
672 function decreaseApproval(
673   address _spender,
674   uint _subtractedValue
675 )
676   public
```

```
  whenNotPaused
  returns (bool success)
{
  return super.decreaseApproval(_spender, _subtractedValue);
}
```

The code meets the specification.

Formal Verification Request 76

PausableToken_decreaseApproval

29, Aug 2019
1.56 ms

Line 668-671 in File VidyCoin.sol

```
/*@CTK PausableToken_decreaseApproval
  @tag assume_completion
  @post paused == false
 */
```

Line 672-681 in File VidyCoin.sol

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

The code meets the specification.

Formal Verification Request 77

BaseERC20Token__burn

29, Aug 2019
270.77 ms

Line 708-713 in File VidyCoin.sol

```
/*@CTK BaseERC20Token__burn
  @tag assume_completion
  @post paused == false
  @post __post.totalSupply_ == totalSupply_ - _value
  @post __post.balances[_from] == balances[_from] - _value
 */
```

Line 714-716 in File VidyCoin.sol

```
function _burn(address _from, uint256 _value) internal whenNotPaused {
  super._burn(_from, _value);
}
```

The code meets the specification.

Formal Verification Request 78

VidyCoin

29, Aug 2019
77.26 ms

Line 723-728 in File VidyCoin.sol

```
723 /*@CTK VidyCoin
724   @tag assume_completion
725   @post __post.totalSupply_ == 1000000000000000000000000000
726   @post __post.balances[msg.sender] == 1000000000000000000000000000
727   @post __post.decimals == 18
728  */
```

Line 729-731 in File VidyCoin.sol

```
729 constructor() BaseERC20Token(1000000000000000000000000000, 18, "VidyCoin", "VIDY")
        public {
730   // nothing else needed
731 }
```

The code meets the specification.

Source Code with CertiK Labels

File TokenTimelock.sol

```
pragma solidity ^0.5.0;

import "./SafeERC20.sol";

/**
 * @title TokenTimelock
 * @dev TokenTimelock is a token holder contract that will allow a
 * beneficiary to extract the tokens after a given release time.
 */
contract TokenTimelock {
    using SafeERC20 for IERC20;

    // ERC20 basic token contract being held
    IERC20 private _token;

    // beneficiary of tokens after they are released
    address private _beneficiary;

    // timestamp when token release is enabled
    uint256 private _releaseTime;

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK TokenTimelock
      @tag assume_completion
      @post releaseTime > block.timestamp
      @post __post._token == token
      @post __post._beneficiary == beneficiary
      @post __post._releaseTime == releaseTime
     */
    constructor (IERC20 token, address beneficiary, uint256 releaseTime) public {
        // solhint-disable-next-line not-rely-on-time
        require(releaseTime > block.timestamp, "TokenTimelock: release time is before
            current time");
        _token = token;
        _beneficiary = beneficiary;
        _releaseTime = releaseTime;
    }

    /**
     * @return the token being held.
     */
    function token() public view returns (IERC20) {
        return _token;
    }

    /**
     * @return the beneficiary of the tokens.
     */
    function beneficiary() public view returns (address) {
        return _beneficiary;
    }

```

```
    /**
     * @return the time when the tokens are released.
     */
    function releaseTime() public view returns (uint256) {
        return _releaseTime;
    }

    /**
     * @notice Transfers tokens held by timelock to beneficiary.
     */
    //$CTK NO_OVERFLOW
    //$CTK NO_BUF_OVERFLOW
    //$CTK NO_ASF
    /*$CTK release
      @tag assume_completion
      @post block.timestamp >= _releaseTime
     */
    function release() public {
        // solhint-disable-next-line not-rely-on-time
        require(block.timestamp >= _releaseTime, "TokenTimelock: current time is before
             release time");

        uint256 amount = _token.balanceOf(address(this));
        require(amount > 0, "TokenTimelock: no tokens to release");

        _token.safeTransfer(_beneficiary, amount);
    }
}
```

File SafeMath.sol

```
pragma solidity ^0.5.0;

/**
 * @dev Wrappers over Solidity's arithmetic operations with added overflow
 * checks.
 *
 * Arithmetic operations in Solidity wrap on overflow. This can easily result
 * in bugs, because programmers usually assume that an overflow raises an
 * error, which is the standard behavior in high level programming languages.
 * `SafeMath` restores this intuition by reverting the transaction when an
 * operation overflows.
 *
 * Using this library instead of the unchecked operations eliminates an entire
 * class of bugs, so it's recommended to use it always.
 */
library SafeMath {
    /**
     * @dev Returns the addition of two unsigned integers, reverting on
     * overflow.
     *
     * Counterpart to Solidity's `+` operator.
     *
     * Requirements:
     * - Addition cannot overflow.
     */
    /*@CTK "SafeMath add"
      @tag spec
      @tag is_pure
```

```
    @post (a + b < a || a + b < b) == __reverted
    @post !__reverted -> __return == a + b
    @post !__reverted -> !__has_overflow
    @post !__reverted -> !__has_assertion_failure
    @post !(__has_buf_overflow)
  */
  function add(uint256 a, uint256 b) internal pure returns (uint256) {
      uint256 c = a + b;
      require(c >= a, "SafeMath: addition overflow");

      return c;
  }

  /**
   * @dev Returns the subtraction of two unsigned integers, reverting on
   * overflow (when the result is negative).
   *
   * Counterpart to Solidity's `-` operator.
   *
   * Requirements:
   * - Subtraction cannot overflow.
   */
  /*@CTK "SafeMath sub"
    @tag spec
    @tag is_pure
    @post (b > a) == __reverted
    @post !__reverted -> __return == a - b
    @post !__reverted -> !__has_overflow
    @post !__reverted -> !__has_assertion_failure
    @post !(__has_buf_overflow)
  */
  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
      require(b <= a, "SafeMath: subtraction overflow");
      uint256 c = a - b;

      return c;
  }

  /**
   * @dev Returns the multiplication of two unsigned integers, reverting on
   * overflow.
   *
   * Counterpart to Solidity's `*` operator.
   *
   * Requirements:
   * - Multiplication cannot overflow.
   */
  /*@CTK "SafeMath mul zero"
    @tag spec
    @tag is_pure
    @pre (a == 0)
    @post __return == 0
  */
  /*@CTK "SafeMath mul nonzero"
    @tag spec
    @tag is_pure
    @pre (a != 0)
    @post (a * b / a != b) == __reverted
```

```
     @post !__reverted -> __return == a * b
     @post !__reverted -> !__has_overflow
     @post !__reverted -> !__has_assertion_failure
     @post !(__has_buf_overflow)
   */
   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
       // Gas optimization: this is cheaper than requiring '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;
       }

       uint256 c = a * b;
       require(c / a == b, "SafeMath: multiplication overflow");

       return c;
   }

   /**
    * @dev Returns the integer division of two unsigned integers. Reverts on
    * division by zero. The result is rounded towards zero.
    *
    * Counterpart to Solidity's `/` operator. Note: this function uses a
    * `revert` opcode (which leaves remaining gas untouched) while Solidity
    * uses an invalid opcode to revert (consuming all remaining gas).
    *
    * Requirements:
    * - The divisor cannot be zero.
    */
   /*@CTK "SafeMath div"
     @tag spec
     @tag is_pure
     @post (b == 0) == __reverted
     @post !__reverted -> __return == a / b
     @post !__reverted -> !__has_overflow
     @post !__reverted -> !__has_assertion_failure
     @post !(__has_buf_overflow)
   */
   function div(uint256 a, uint256 b) internal pure returns (uint256) {
       // Solidity only automatically asserts when dividing by 0
       require(b > 0, "SafeMath: division by zero");
       uint256 c = a / b;
       // assert(a == b * c + a % b); // There is no case in which this doesn't hold

       return c;
   }

   /**
    * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer
        modulo),
    * Reverts when dividing by zero.
    *
    * Counterpart to Solidity's `%` operator. This function uses a `revert`
    * opcode (which leaves remaining gas untouched) while Solidity uses an
    * invalid opcode to revert (consuming all remaining gas).
    *
    * Requirements:
```

```
     * - The divisor cannot be zero.
     */
    /*@CTK "SafeMath mod"
      @tag spec
      @tag is_pure
      @post (b == 0) == __reverted
      @post !__reverted -> __return == a % b
      @post !__reverted -> !__has_overflow
      @post !__reverted -> !__has_assertion_failure
      @post !(__has_buf_overflow)
    */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0, "SafeMath: modulo by zero");
        return a % b;
    }
}
```

File VidyCoin.sol

```
/**
 *Submitted for verification at Etherscan.io on 2019-01-16
*/

pragma solidity ^0.4.23;

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

```
    @post !__reverted -> !(__has_assertion_failure)
    @post !(__has_buf_overflow)
  */
  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 !__reverted -> !(__has_assertion_failure)
    @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;
  }
}

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 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 Ownable
 * @dev The Ownable contract has an owner address, and provides basic authorization control
 * functions, this simplifies the implementation of "user permissions".
 */
contract Ownable {
  address public owner;


  event OwnershipRenounced(address indexed previousOwner);
  event OwnershipTransferred(
    address indexed previousOwner,
    address indexed newOwner
  );


  /**
   * @dev The Ownable constructor sets the original `owner` of the contract to the sender
   * account.
   */
  constructor() public {
    owner = msg.sender;
  }

  /**
   * @dev Throws if called by any account other than the owner.
   */
  modifier onlyOwner() {
    require(msg.sender == owner);
    _;
  }

  /**
   * @dev Allows the current owner to relinquish control of the contract.
   */
  /*@CTK renounceOwnership
    @tag assume_completion
    @post msg.sender == owner
    @post __post.owner == address(0)
   */
  function renounceOwnership() public onlyOwner {
    emit OwnershipRenounced(owner);
    owner = address(0);
  }
```

```

  /**
   * @dev Allows the current owner to transfer control of the contract to a newOwner.
   * @param _newOwner The address to transfer ownership to.
   */
  function transferOwnership(address _newOwner) public onlyOwner {
    _transferOwnership(_newOwner);
  }

  /**
   * @dev Transfers control of the contract to a newOwner.
   * @param _newOwner The address to transfer ownership to.
   */
  /*@CTK transferOwnership
    @tag assume_completion
    @post _newOwner != address(0)
    @post __post.owner == _newOwner
   */
  function _transferOwnership(address _newOwner) internal {
    require(_newOwner != address(0));
    emit OwnershipTransferred(owner, _newOwner);
    owner = _newOwner;
  }
}

/**
 * @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 NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK totalSupply
    @tag assume_completion
    @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.
  */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK FAIL NO_ASF
```

```
  /*@CTK transfer
    @tag assume_completion
    @pre _to != address(0)
    @pre _value <= balances[msg.sender]
    @post (msg.sender != _to) -> (__post.balances[_to] == balances[_to] + _value)
    @post (msg.sender != _to) -> (__post.balances[msg.sender] == balances[msg.sender]
        - _value)
    @post (msg.sender == _to) -> (__post.balances[_to] == balances[_to])
    @post (msg.sender == _to) -> (__post.balances[msg.sender] == balances[msg.sender])
    @post __return == true
  */
  function transfer(address _to, uint256 _value) public returns (bool) {
    require(_to != address(0));
    require(_value <= balances[msg.sender]);

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

  /**
  * @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 NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK balanceOf
    @tag assume_completion
    @post (__return) == (balances[_owner])
  */
  function balanceOf(address _owner) public view returns (uint256) {
    return balances[_owner];
  }

}

/**
 * @title Burnable Token
 * @dev Token that can be irreversibly burned (destroyed).
 */
contract BurnableToken is BasicToken {

  event Burn(address indexed burner, uint256 value);

  /**
   * @dev Burns a specific amount of tokens.
   * @param _value The amount of token to be burned.
   */

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK FAIL NO_ASF
  /*@CTK burn
    @tag assume_completion
    @post (_value <= balances[msg.sender])
```

```
    @post (__post.totalSupply_) == ((totalSupply_) - (_value))
    @post (__post.balances[msg.sender]) == ((balances[msg.sender]) - (_value))
   */
  function burn(uint256 _value) public {
    _burn(msg.sender, _value);
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK FAIL NO_ASF
  /*@CTK _burn
    @tag assume_completion
    @post (_value <= balances[_who])
    @post (__post.totalSupply_) == ((totalSupply_) - (_value))
    @post (__post.balances[_who]) == ((balances[_who]) - (_value))
  */
  function _burn(address _who, uint256 _value) internal {
    require(_value <= balances[_who]);
    // no need to require value <= totalSupply, since that would imply the
    // sender's balance is greater than the totalSupply, which *should* be an
        assertion failure

    balances[_who] = balances[_who].sub(_value);
    totalSupply_ = totalSupply_.sub(_value);
    emit Burn(_who, _value);
    emit Transfer(_who, address(0), _value);
  }
}





/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * @dev https://github.com/ethereum/EIPs/issues/20
 * @dev 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
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK FAIL NO_ASF
  /*@CTK "transferFrom correctness"
```

```
    @tag assume_completion
    @post _to != 0x0
    @post _value <= balances[_from] && _value <= allowed[_from][msg.sender]
    @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)
  {
    require(_to != address(0));
    require(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

    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.
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "approve correctness"
    @post __post.allowed[msg.sender][_spender] == _value
   */
  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
```

```
     .
 */
//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "allowance correctness"
  @post __return == allowed[_owner][_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.
 */

//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK FAIL NO_ASF
/*@CTK "increaseApproval correctness"
  @tag assume_completion
  @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
      _addedValue
 */
function increaseApproval(
  address _spender,
  uint _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.
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK decreaseApproval0
    @pre __return == true
    @pre allowed[msg.sender][_spender] <= _subtractedValue
    @post __post.allowed[msg.sender][_spender] == 0
   */
  /*@CTK decreaseApproval
    @pre __return == true
    @pre allowed[msg.sender][_spender] > _subtractedValue
    @post __post.allowed[msg.sender][_spender] ==
        allowed[msg.sender][_spender] - _subtractedValue
   */
  function decreaseApproval(
    address _spender,
    uint _subtractedValue
  )
    public
    returns (bool)
  {
    uint 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;
  }

}



/**
 * @title Standard Burnable Token
 * @dev Adds burnFrom method to ERC20 implementations
 */
contract StandardBurnableToken is BurnableToken, StandardToken {

  /**
   * @dev Burns a specific amount of tokens from the target address and decrements
       allowance
   * @param _from address The address which you want to send tokens from
   * @param _value uint256 The amount of token to be burned
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK FAIL NO_ASF
  /*@CTK burnFrom
    @tag assume_completion
    @post (_value <= allowed[_from][msg.sender])
    @post (_value <= balances[_from])
    @post (__post.allowed[_from][msg.sender]) == ((allowed[_from][msg.sender]) - (
```

```
      _value))
    @post (__post.balances[_from]) == ((balances[_from]) - (_value))
    @post __post.totalSupply_ == (totalSupply_ - _value)
   */
  function burnFrom(address _from, uint256 _value) public {
    require(_value <= allowed[_from][msg.sender]);
    // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
      ,
    // this function needs to emit an event with the updated approval.
    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
    _burn(_from, _value);
  }
}









/**
 * @title DetailedERC20 token
 * @dev The decimals are only for visualization purposes.
 * All the operations are done using the smallest and indivisible token unit,
 * just as on Ethereum all the operations are done in wei.
 */
contract DetailedERC20 is ERC20 {
  string public name;
  string public symbol;
  uint8 public decimals;

  /*@CTK DetailedERC20
    @tag assume_completion
    @post __post.name == _name
    @post __post.symbol == _symbol
    @post __post.decimals == _decimals
   */
  constructor(string _name, string _symbol, uint8 _decimals) public {
    name = _name;
    symbol = _symbol;
    decimals = _decimals;
  }
}


/**
 * @title Pausable
 * @dev Base contract which allows children to implement an emergency stop mechanism.
 */
contract Pausable is Ownable {
  event Pause();
  event Unpause();

```

```
  bool public paused = false;


  /**
   * @dev Modifier to make a function callable only when the contract is not paused.
   */
  modifier whenNotPaused() {
    require(!paused);
    _;
  }

  /**
   * @dev Modifier to make a function callable only when the contract is paused.
   */
  modifier whenPaused() {
    require(paused);
    _;
  }

  /**
   * @dev called by the owner to pause, triggers stopped state
   */
  /*@CTK pause
    @tag assume_completion
    @post paused == false
    @post owner == msg.sender
    @post __post.paused == true
   */
  function pause() onlyOwner whenNotPaused public {
    paused = true;
    emit Pause();
  }

  /**
   * @dev called by the owner to unpause, returns to normal state
   */
  /*@CTK unpause
    @tag assume_completion
    @post paused == true
    @post owner == msg.sender
    @post __post.paused == false
   */
  function unpause() onlyOwner whenPaused public {
    paused = false;
    emit Unpause();
  }
}



/**
 * @title Pausable token
 * @dev StandardToken modified with pausable transfers.
 **/
contract PausableToken is StandardToken, Pausable {

  /*@CTK PausableToken_transfer
    @tag assume_completion
```

```
  @post paused == false
 */
function transfer(
  address _to,
  uint256 _value
)
  public
  whenNotPaused
  returns (bool)
{
  return super.transfer(_to, _value);
}

/*@CTK PausableToken_transferFrom
  @tag assume_completion
  @post paused == false
 */
function transferFrom(
  address _from,
  address _to,
  uint256 _value
)
  public
  whenNotPaused
  returns (bool)
{
  return super.transferFrom(_from, _to, _value);
}

/*@CTK PausableToken_approve
  @tag assume_completion
  @post paused == false
 */
function approve(
  address _spender,
  uint256 _value
)
  public
  whenNotPaused
  returns (bool)
{
  return super.approve(_spender, _value);
}

/*@CTK PausableToken_decreaseApproval
  @tag assume_completion
  @post paused == false
 */
function increaseApproval(
  address _spender,
  uint _addedValue
)
  public
  whenNotPaused
  returns (bool success)
{
  return super.increaseApproval(_spender, _addedValue);
}
```

```
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK PausableToken_decreaseApproval
    @tag assume_completion
    @post paused == false
   */
  function decreaseApproval(
    address _spender,
    uint _subtractedValue
  )
    public
    whenNotPaused
    returns (bool success)
  {
    return super.decreaseApproval(_spender, _subtractedValue);
  }
}


/**
 * A base token for the VidyCoin (or any other coin with similar behavior).
 * Compatible with contracts and UIs expecting a ERC20 token.
 * Provides also a convenience method to burn tokens, permanently removing them from
    the pool;
 * the intent of this convenience method is for users who wish to burn tokens
 * (as they always can via a transfer to an unowned address or self-destructing
 * contract) to do so in a way that is then reflected in the token's total supply.
 */
contract BaseERC20Token is StandardBurnableToken, PausableToken, DetailedERC20 {

  constructor(
    uint256 _initialAmount,
    uint8 _decimalUnits,
    string _tokenName,
    string _tokenSymbol
  ) DetailedERC20(_tokenName, _tokenSymbol, _decimalUnits) public {
    totalSupply_ = _initialAmount;
    balances[msg.sender] = totalSupply_;
  }

  // override the burnable token's "Burn" function: don't allow tokens to
  // be burned when paused

  /*@CTK BaseERC20Token__burn
    @tag assume_completion
    @post paused == false
    @post __post.totalSupply_ == totalSupply_ - _value
    @post __post.balances[_from] == balances[_from] - _value
   */
  function _burn(address _from, uint256 _value) internal whenNotPaused {
    super._burn(_from, _value);
  }

}
```

```
contract VidyCoin is BaseERC20Token {

  /*@CTK VidyCoin
    @tag assume_completion
    @post __post.totalSupply_ == 1000000000000000000000000000
    @post __post.balances[msg.sender] == 1000000000000000000000000000
    @post __post.decimals == 18
   */
  constructor() BaseERC20Token(1000000000000000000000000000, 18, "VidyCoin", "VIDY")
      public {
    // nothing else needed
  }
}
```