CertiK-Tunwu Audit Report For ONEDS

Request Date: 2019-06-01
Revision Date: 2019-06-10
Platform Name: Ethereum

TUNWU

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-Tunwu and ONEDS(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-Tunwu'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 1.4B in assets.

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

Exective Summary

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

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

Vulnerability Classification

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

Critical

The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.

Medium

The code implementation does not match the specification at certain condition, or it could affect the security standard by lost of access control.

Low

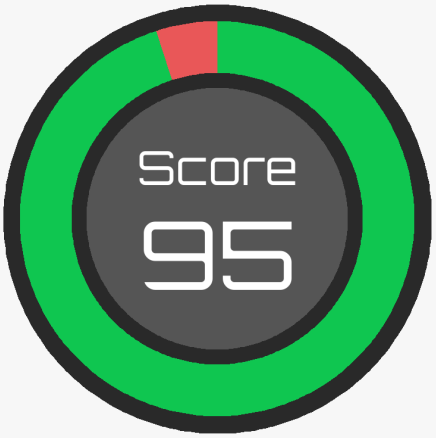
The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerability, but no concern found yet.

Testing Summary

PASS

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

Jun 10, 2019



Type of Issues

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

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

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

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

No issue found. (Notice: The warning signs for `mul()`, `div()`, `sub()`, `add()` of the `SafeMath` library are internal results of CertiK's analysis engine and are not indication of error in the client's code.)

Manual Review Notes

Source Code SHA-256 Checksum

- **ONEDS.sol** `68e2aa54652c231abb5c90a29a74a5845b000e8a692e064c95543a4914a14c00`

Summary

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

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

Overall we found the `OneDSToken` contracts follows good practices, with a reasonable amount of features on top of the ERC20 related to administrative controls by the token issuer. With the final update of source code and delivery of the audit report, we conclude that the contract is not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend seeking multiple opinions, more test coverage, and sandbox deployments before the mainnet release.

Best practice

The checklist below helps to reflect the general quality of the solidity project.

Solidity Protocol

✓ Use latest stable solidity version

✓ Handle possible errors properly when making external calls

✓ Provide error message along with require()

✓ Use modifiers properly

✓ Use events to monitor contract activities

✓ Refer and use libraries properly

✓ No compiler warnings

Privilege Control

✓ Provide stop functionality for control and emergency handling

✓ Restrict access to sensitive functions

Documentation

– Provide project documentation and execution guidance

✓ Provide inline comment for function intention

– Provide instruction to initialize and execute the test files

Testing

– Provide migration scripts

– Provide test scripts and coverage for potential scenarios

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 1 in File OneDSToken.sol

```
1 pragma solidity ^0.4.13;
```

Version to compile has the following bug: 0.4.13: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue, ECRecoverMalformedInput 0.4.14: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector, DelegateCallReturnValue 0.4.15: UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector 0.4.16: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, NestedArrayFunctionCallDecoder, ZeroFunctionSelector 0.4.17: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder, ZeroFunctionSelector 0.4.18: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.19: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.20: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.21: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.22: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, OneOfTwoConstructorsSkipped 0.4.23: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

INSECURE_COMPILER_VERSION

Line 1 in File Migrations.sol

```
1 pragma solidity ^0.4.17;
```

Version to compile has the following bug: 0.4.17: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder, ZeroFunctionSelector 0.4.18: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.19: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.20: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.21: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, NestedArrayFunctionCallDecoder 0.4.22: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData, OneOfTwoConstructorsSkipped 0.4.23: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

Formal Verification Request 1

Method will not encounter an assertion failure.

10, Jun 2019
26.9 ms

Line 4 in File OneDSToken.sol

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

Line 12-19 in File OneDSToken.sol

```
12  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
13    if (a == 0) {
14      return 0;
15    }
16    uint256 c = a * b;
17    assert(c / a == b);
18    return c;
19  }
```

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        __return = 0
21        block = {
22          "number": 0,
23          "timestamp": 0
24        }
25    }
26    Address_Map = [
27      {
28        "key": "ALL_OTHERS",
29        "value": "EmptyAddress"
30      }
31    ]
32
33 Function invocation is reverted.
```

Formal Verification Request 2

SafeMath mul

10, Jun 2019
347.82 ms

Line 5-11 in File OneDSToken.sol

```
5   /*@CTK "SafeMath mul"
6     @post ((a > 0) && (((a * b) / a) != b)) == (__reverted)
7     @post !__reverted -> __return == a * b
8     @post !__reverted == !__has_overflow
9     @post !__reverted -> !(__has_assertion_failure)
10    @post !(__has_buf_overflow)
11    */
```

Line 12-19 in File OneDSToken.sol

```
12  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
13    if (a == 0) {
14      return 0;
15    }
16    uint256 c = a * b;
17    assert(c / a == b);
18    return c;
19  }
```

The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.

10, Jun 2019
5.46 ms

Line 20 in File OneDSToken.sol

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

Line 28-33 in File OneDSToken.sol

```
28  function div(uint256 a, uint256 b) internal pure returns (uint256) {
29    // assert(b > 0); // Solidity automatically throws when dividing by 0
30    uint256 c = a / b;
31    // assert(a == b * c + a % b); // There is no case in which this doesn't hold
32    return c;
33  }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3     Input = {
4         a = 0
5         b = 0
6     }
7     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 4

SafeMath div

10, Jun 2019
0.85 ms

Line 21-27 in File OneDSToken.sol

```
/*@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)
*/
```

Line 28-33 in File OneDSToken.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 c;
}
```

The code meets the specification.

Formal Verification Request 5

Method will not encounter an assertion failure.

10, Jun 2019
11.26 ms

Line 34 in File OneDSToken.sol

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

Line 42-45 in File OneDSToken.sol

```
42  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
43    assert(b <= a);
44    return a - b;
45  }
```

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 = [
27      {
28        "key": "ALL_OTHERS",
29        "value": "EmptyAddress"
30      }
31    ]
32
33 Function invocation is reverted.
```

Formal Verification Request 6

SafeMath sub

10, Jun 2019

0.85 ms

Line 35-41 in File OneDSToken.sol

```
35  /*@CTK "SafeMath sub"
36    @post (a < b) == __reverted
37    @post !__reverted -> __return == a - b
38    @post !__reverted -> !__has_overflow
39    @post !__reverted -> !(__has_assertion_failure)
40    @post !(__has_buf_overflow)
41  */
```

Line 42-45 in File OneDSToken.sol

```
42  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
43    assert(b <= a);
44    return a - b;
45  }
```

The code meets the specification.

Formal Verification Request 7

Method will not encounter an assertion failure.

10, Jun 2019
11.45 ms

Line 46 in File OneDSToken.sol

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

Line 54-58 in File OneDSToken.sol

```
54  function add(uint256 a, uint256 b) internal pure returns (uint256) {
55    uint256 c = a + b;
56    assert(c >= a);
57    return c;
58  }
```

This code violates the specification.

```
1  Counter Example:
2  Before Execution:
3     Input = {
4         a = 13
5         b = 243
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        }
```

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

Function invocation is reverted.
```

Formal Verification Request 8

SafeMath add

10, Jun 2019
2.38 ms

Line 47-53 in File OneDSToken.sol

```
/*@CTK "SafeMath add"
  @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)
*/
```

Line 54-58 in File OneDSToken.sol

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

The code meets the specification.

Formal Verification Request 9

Ownable

10, Jun 2019
4.19 ms

Line 69-71 in File OneDSToken.sol

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

Line 72-74 in File OneDSToken.sol

```
72  function Ownable() public {
73    owner = msg.sender;
74  }
```

The code meets the specification.

Formal Verification Request 10

transferOwnership

10, Jun 2019
20.19 ms

Line 90-95 in File OneDSToken.sol

```
90  /*@CTK transferOwnership
91    @tag assume_completion
92    @post (msg.sender == owner)
93    @post newOwner != address(0)
94    @post __post.owner == newOwner
95   */
```

Line 96-100 in File OneDSToken.sol

```
96  function transferOwnership(address newOwner) public onlyOwner {
97    require(newOwner != address(0));
98    emit OwnershipTransferred(owner, newOwner);
99    owner = newOwner;
100 }
```

The code meets the specification.

Formal Verification Request 11

Ownable

10, Jun 2019
19.54 ms

Line 129-134 in File OneDSToken.sol

```
129 /*@CTK Ownable
130   @tag assume_completion
131   @pre !paused
132   @post msg.sender == owner
133   @post __post.paused
134  */
```

Line 135-138 in File OneDSToken.sol

```
135 function pause() onlyOwner whenNotPaused public {
136   paused = true;
137   emit Pause();
138 }
```

The code meets the specification.

Formal Verification Request 12

Ownable

10, Jun 2019
22.49 ms

Line 143-148 in File OneDSToken.sol

```
143  /*@CTK Ownable
144    @tag assume_completion
145    @pre paused
146    @post msg.sender == owner
147    @post !(__post.paused)
148   */
```

Line 149-152 in File OneDSToken.sol

```
149  function unpause() onlyOwner whenPaused public {
150    paused = false;
151    emit Unpause();
152  }
```

The code meets the specification.

Formal Verification Request 13

transfer correctness

10, Jun 2019
239.96 ms

Line 178-191 in File OneDSToken.sol

```
178  /*@CTK "transfer correctness"
179    @tag assume_completion
180    @pre balances[msg.sender] + balances[_to] >= balances[msg.sender]
181    @pre balances[msg.sender] + balances[_to] >= balances[_to]
182    @post _to != address(0)
183    @post _value <= balances[msg.sender]
184    @post balances[_to] + _value >= balances[_to]
185    @post _to != msg.sender -> __post.balances[msg.sender] == balances[msg.sender] -
           _value
186    @post _to != msg.sender -> __post.balances[_to] == balances[_to] + _value
187    @post _to == msg.sender -> __post.balances[msg.sender] == balances[msg.sender]
188    @post !(__has_overflow)
189    @post !(__has_buf_overflow)
190    @post !(__has_assertion_failure)
191    */
```

Line 192-201 in File OneDSToken.sol

```
192  function transfer(address _to, uint256 _value) public returns (bool) {
193    require(_to != address(0));
194    require(_value <= balances[msg.sender]);
195
196    // SafeMath.sub will throw if there is not enough balance.
197    balances[msg.sender] = balances[msg.sender].sub(_value);
198    balances[_to] = balances[_to].add(_value);
```

```
199     emit Transfer(msg.sender, _to, _value);
200     return true;
201   }
```

The code meets the specification.

Formal Verification Request 14

transferFrom correctness

10, Jun 2019
538.08 ms

Line 223-235 in File OneDSToken.sol

```
223   /*@CTK "transferFrom correctness"
224     @tag assume_completion
225     @post _to != address(0)
226     @post _value <= balances[_from]
227     @post _value <= allowed[_from][msg.sender]
228     @post _from != _to -> __post.balances[_from] == balances[_from] - _value
229     @post _from != _to -> __post.balances[_to] == balances[_to] + _value
230     @post _from == _to -> __post.balances[_from] == balances[_from]
231     @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
232     @post !(__has_overflow)
233     @post !(__has_buf_overflow)
234     @post !(__has_assertion_failure)
235     */
```

Line 236-246 in File OneDSToken.sol

```
236   function transferFrom(address _from, address _to, uint256 _value) public returns (
          bool) {
237     require(_to != address(0));
238     require(_value <= balances[_from]);
239     require(_value <= allowed[_from][msg.sender]);
240
241     balances[_from] = balances[_from].sub(_value);
242     balances[_to] = balances[_to].add(_value);
243     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
244     emit Transfer(_from, _to, _value);
245     return true;
246   }
```

The code meets the specification.

Formal Verification Request 15

If method completes, integer overflow would not happen.

10, Jun 2019
7.58 ms

Line 253 in File OneDSToken.sol

```
253   //@CTK NO_OVERFLOW
```

Line 260-264 in File OneDSToken.sol

```
260  function approve(address _spender, uint256 _value) public returns (bool) {
261    allowed[msg.sender][_spender] = _value;
262    emit Approval(msg.sender, _spender, _value);
263    return true;
264  }
```

The code meets the specification.

Formal Verification Request 16

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

10, Jun 2019
0.29 ms

Line 254 in File OneDSToken.sol

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

Line 260-264 in File OneDSToken.sol

```
260  function approve(address _spender, uint256 _value) public returns (bool) {
261    allowed[msg.sender][_spender] = _value;
262    emit Approval(msg.sender, _spender, _value);
263    return true;
264  }
```

The code meets the specification.

Formal Verification Request 17

Method will not encounter an assertion failure.

10, Jun 2019
0.28 ms

Line 255 in File OneDSToken.sol

```
255  //@CTK NO_ASF
```

Line 260-264 in File OneDSToken.sol

```
260  function approve(address _spender, uint256 _value) public returns (bool) {
261    allowed[msg.sender][_spender] = _value;
262    emit Approval(msg.sender, _spender, _value);
263    return true;
264  }
```

The code meets the specification.

Formal Verification Request 18

approve correctness

10, Jun 2019
1.06 ms

Line 256-259 in File OneDSToken.sol

```
256  /*@CTK "approve correctness"
257    @post __post.allowed[msg.sender][_spender] == _value
258    @post __return
259    */
```

Line 260-264 in File OneDSToken.sol

```
260  function approve(address _spender, uint256 _value) public returns (bool) {
261    allowed[msg.sender][_spender] = _value;
262    emit  Approval(msg.sender, _spender, _value);
263    return true;
264  }
```

The code meets the specification.

Formal Verification Request 19

allowance

10, Jun 2019
4.45 ms

Line 272-274 in File OneDSToken.sol

```
272  /*@CTK allowance
273    @post __return == allowed[_owner][_spender]
274   */
```

Line 275-277 in File OneDSToken.sol

```
275  function allowance(address _owner, address _spender) public view returns (uint256) {
276    return allowed[_owner][_spender];
277  }
```

The code meets the specification.

Formal Verification Request 20

increaseApproval

10, Jun 2019
27.83 ms

Line 287-290 in File OneDSToken.sol

```
287  /*@CTK increaseApproval
288    @tag assume_completion
289    @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
           _addedValue
290   */
```

Line 291-295 in File OneDSToken.sol

```
291  function increaseApproval(address _spender, uint _addedValue) public returns (bool)
         {
292    allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
293    emit  Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
294    return true;
295  }
```

The code meets the specification.

Formal Verification Request 21

decreaseApproval

10, Jun 2019
42.03 ms

Line 305-309 in File OneDSToken.sol

```
305  /*@CTK decreaseApproval
306    @tag assume_completion
307    @post (_subtractedValue > allowed[msg.sender][_spender]) -> __post.allowed[msg.
           sender][_spender] == 0
308    @post (_subtractedValue <= allowed[msg.sender][_spender]) -> __post.allowed[msg.
           sender][_spender] == allowed[msg.sender][_spender] - _subtractedValue
309   */
```

Line 310-319 in File OneDSToken.sol

```
310  function decreaseApproval(address _spender, uint _subtractedValue) public returns (
         bool) {
311    uint oldValue = allowed[msg.sender][_spender];
312    if (_subtractedValue > oldValue) {
313      allowed[msg.sender][_spender] = 0;
314    } else {
315      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
316    }
317  emit  Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
318    return true;
319  }
```

The code meets the specification.

Formal Verification Request 22

OneDSToken

10, Jun 2019
11.36 ms

Line 355-361 in File OneDSToken.sol

```
355    /*@CTK OneDSToken
356      @tag assume_completion
357      @post __post.balances[msg.sender] == __post.totalSupply
358      @post !(__has_overflow)
```

```
359      @post !(__has_buf_overflow)
360      @post !(__has_assertion_failure)
361     */
```

Line 362-365 in File OneDSToken.sol

```
362    function OneDSToken() public {
363        totalSupply = INITIAL_SUPPLY;
364        balances[msg.sender] = INITIAL_SUPPLY;
365    }
```

The code meets the specification.

Formal Verification Request 23

If method completes, integer overflow would not happen.

10, Jun 2019
5.06 ms

Line 11 in File Migrations.sol

```
11    //@CTK NO_OVERFLOW
```

Line 26-28 in File Migrations.sol

```
26    function Migrations() public {
27      owner = msg.sender;
28    }
```

The code meets the specification.

Formal Verification Request 24

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

10, Jun 2019
0.34 ms

Line 12 in File Migrations.sol

```
12    //@CTK NO_BUF_OVERFLOW
```

Line 26-28 in File Migrations.sol

```
26    function Migrations() public {
27      owner = msg.sender;
28    }
```

The code meets the specification.

Formal Verification Request 25

Method will not encounter an assertion failure.

10, Jun 2019
0.38 ms

Line 13 in File Migrations.sol

```
13    //@CTK NO_ASF
```

Line 26-28 in File Migrations.sol

```
26    function Migrations() public {
27      owner = msg.sender;
28    }
```

The code meets the specification.

Formal Verification Request 26

restricted

10, Jun 2019
1.42 ms

Line 14-16 in File Migrations.sol

```
14    /*@CTK "restricted"
15      @post (__post.owner) == (msg.sender)
16    */
```

Line 26-28 in File Migrations.sol

```
26    function Migrations() public {
27      owner = msg.sender;
28    }
```

The code meets the specification.

Formal Verification Request 27

migrations

10, Jun 2019
0.8 ms

Line 17-19 in File Migrations.sol

```
17    /*@CTK "migrations"
18      @post (__post.owner) == (msg.sender)
19    */
```

Line 26-28 in File Migrations.sol

```
26    function Migrations() public {
27      owner = msg.sender;
28    }
```

The code meets the specification.

Formal Verification Request 28

init_migrations

10, Jun 2019
0.86 ms

Line 20-22 in File Migrations.sol

```
20  /*@CTK "init_migrations"
21    @post (__post.owner) == (msg.sender)
22  */
```

Line 26-28 in File Migrations.sol

```
26  function Migrations() public {
27    owner = msg.sender;
28  }
```

The code meets the specification.

Formal Verification Request 29

Migrations

10, Jun 2019
0.79 ms

Line 23-25 in File Migrations.sol

```
23  /*@CTK "Migrations"
24    @post (__post.owner) == (msg.sender)
25  */
```

Line 26-28 in File Migrations.sol

```
26  function Migrations() public {
27    owner = msg.sender;
28  }
```

The code meets the specification.

Formal Verification Request 30

If method completes, integer overflow would not happen.

10, Jun 2019
7.21 ms

Line 30 in File Migrations.sol

```
30  //@CTK NO_OVERFLOW
```

Line 44-46 in File Migrations.sol

```
44  function setCompleted(uint completed) public restricted {
45    last_completed_migration = completed;
46  }
```

The code meets the specification.

Formal Verification Request 31

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

10, Jun 2019
0.38 ms

Line 31 in File Migrations.sol

```
31    //@CTK NO_BUF_OVERFLOW
```

Line 44-46 in File Migrations.sol

```
44    function setCompleted(uint completed) public restricted {
45      last_completed_migration = completed;
46    }
```

The code meets the specification.

Formal Verification Request 32

Method will not encounter an assertion failure.

10, Jun 2019
0.34 ms

Line 32 in File Migrations.sol

```
32    //@CTK NO_ASF
```

Line 44-46 in File Migrations.sol

```
44    function setCompleted(uint completed) public restricted {
45      last_completed_migration = completed;
46    }
```

The code meets the specification.

Formal Verification Request 33

set_complete

10, Jun 2019
1.43 ms

Line 33-36 in File Migrations.sol

```
33    /*@CTK "set_complete"
34      @pre (msg.sender) == (owner)
35      @post (__post.last_completed_migration) == (completed)
36    */
```

Line 44-46 in File Migrations.sol

```
44    function setCompleted(uint completed) public restricted {
45      last_completed_migration = completed;
46    }
```

The code meets the specification.

Formal Verification Request 34

setCompleted

10, Jun 2019
1.19 ms

Line 37-40 in File Migrations.sol

```
37    /*@CTK "setCompleted"
38      @pre (msg.sender) == (owner)
39      @post (__post.last_completed_migration) == (completed)
40    */
```

Line 44-46 in File Migrations.sol

```
44    function setCompleted(uint completed) public restricted {
45      last_completed_migration = completed;
46    }
```

The code meets the specification.

Formal Verification Request 35

setCompleted

10, Jun 2019
1.19 ms

Line 41-43 in File Migrations.sol

```
41    /*@CTK "setCompleted"
42      @post ((__post.last_completed_migration) != (completed)) -> ((msg.sender) != (
            owner))
43    */
```

Line 44-46 in File Migrations.sol

```
44    function setCompleted(uint completed) public restricted {
45      last_completed_migration = completed;
46    }
```

The code meets the specification.

Formal Verification Request 36

If method completes, integer overflow would not happen.

10, Jun 2019
24.54 ms

Line 48 in File Migrations.sol

```
48    //@CTK NO_OVERFLOW
```

Line 51-54 in File Migrations.sol

```
51    function upgrade(address new_address) public restricted {
52      Migrations upgraded = Migrations(new_address);
53      upgraded.setCompleted(last_completed_migration);
54    }
```

The code meets the specification.

Formal Verification Request 37

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

10, Jun 2019
0.52 ms

Line 49 in File Migrations.sol

```
49    //@CTK NO_BUF_OVERFLOW
```

Line 51-54 in File Migrations.sol

```
51    function upgrade(address new_address) public restricted {
52      Migrations upgraded = Migrations(new_address);
53      upgraded.setCompleted(last_completed_migration);
54    }
```

The code meets the specification.

Formal Verification Request 38

Method will not encounter an assertion failure.

10, Jun 2019
0.51 ms

Line 50 in File Migrations.sol

```
50    //@CTK NO_ASF
```

Line 51-54 in File Migrations.sol

```
51    function upgrade(address new_address) public restricted {
52      Migrations upgraded = Migrations(new_address);
53      upgraded.setCompleted(last_completed_migration);
54    }
```

The code meets the specification.

Source Code with CertiK-Tunwu Labels

File OneDSToken.sol

```
pragma solidity ^0.4.13;

library SafeMath {
  //@CTK FAIL NO_ASF
  /*@CTK "SafeMath mul"
    @post ((a > 0) && (((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) {
    if (a == 0) {
      return 0;
    }
    uint256 c = a * b;
    assert(c / a == b);
    return c;
  }
  //@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 c;
  }
  //@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;
  }
  //@CTK FAIL NO_ASF
  /*@CTK "SafeMath add"
    @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) {
```

```
55      uint256 c = a + b;
56      assert(c >= a);
57      return c;
58    }
59  }
60
61  contract Ownable {
62    address public owner;
63
64    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
65
66    /**
67     * @dev The Ownable constructor sets the original `owner` of the contract to the
          sender
68     */
69    /*@CTK Ownable
70      @post __post.owner == msg.sender
71     */
72    function Ownable() public {
73      owner = msg.sender;
74    }
75
76
77    /**
78     * @dev Throws if called by any account other than the owner.
79     */
80    modifier onlyOwner() {
81      require(msg.sender == owner);
82      _;
83    }
84
85
86    /**
87     * @dev Allows the current owner to transfer control of the contract to a newOwner.
88     * @param newOwner The address to transfer ownership to.
89     */
90    /*@CTK transferOwnership
91      @tag assume_completion
92      @post (msg.sender == owner)
93      @post newOwner != address(0)
94      @post __post.owner == newOwner
95     */
96    function transferOwnership(address newOwner) public onlyOwner {
97      require(newOwner != address(0));
98      emit OwnershipTransferred(owner, newOwner);
99      owner = newOwner;
100   }
101
102 }
103
104 contract Pausable is Ownable {
105   event Pause();
106   event Unpause();
107
108   bool public paused = false;
109
110   /**
111    * @dev Modifier to make a function callable only when the contract is not paused.
```

```
112     */
113   modifier whenNotPaused() {
114     require(!paused);
115     _;
116   }
117
118   /**
119    * @dev Modifier to make a function callable only when the contract is paused.
120    */
121   modifier whenPaused() {
122     require(paused);
123     _;
124   }
125
126   /**
127    * @dev called by the owner to pause, triggers stopped state
128    */
129   /*@CTK Ownable
130     @tag assume_completion
131     @pre !paused
132     @post msg.sender == owner
133     @post __post.paused
134    */
135   function pause() onlyOwner whenNotPaused public {
136     paused = true;
137     emit  Pause();
138   }
139
140   /**
141    * @dev called by the owner to unpause, returns to normal state
142    */
143   /*@CTK Ownable
144     @tag assume_completion
145     @pre paused
146     @post msg.sender == owner
147     @post !(__post.paused)
148    */
149   function unpause() onlyOwner whenPaused public {
150     paused = false;
151     emit Unpause();
152   }
153 }
154
155 contract ERC20Basic {
156   uint256 public totalSupply;
157   function balanceOf(address who) public view returns (uint256);
158   function transfer(address to, uint256 value) public returns (bool);
159   event Transfer(address indexed from, address indexed to, uint256 value);
160 }
161
162 contract ERC20 is ERC20Basic {
163   function allowance(address owner, address spender) public view returns (uint256);
164   function transferFrom(address from, address to, uint256 value) public returns (bool)
        ;
165   function approve(address spender, uint256 value) public returns (bool);
166   event Approval(address indexed owner, address indexed spender, uint256 value);
167 }
168
```

```
169 contract BasicToken is ERC20Basic {
170   using SafeMath for uint256;
171   mapping(address => uint256) balances;
172
173   /**
174   * @dev transfer token for a specified address
175   * @param _to The address to transfer to.
176   * @param _value The amount to be transferred.
177   */
178   /*@CTK "transfer correctness"
179     @tag assume_completion
180     @pre balances[msg.sender] + balances[_to] >= balances[msg.sender]
181     @pre balances[msg.sender] + balances[_to] >= balances[_to]
182     @post _to != address(0)
183     @post _value <= balances[msg.sender]
184     @post balances[_to] + _value >= balances[_to]
185     @post _to != msg.sender -> __post.balances[msg.sender] == balances[msg.sender] -
          _value
186     @post _to != msg.sender -> __post.balances[_to] == balances[_to] + _value
187     @post _to == msg.sender -> __post.balances[msg.sender] == balances[msg.sender]
188     @post !(__has_overflow)
189     @post !(__has_buf_overflow)
190     @post !(__has_assertion_failure)
191     */
192   function transfer(address _to, uint256 _value) public returns (bool) {
193     require(_to != address(0));
194     require(_value <= balances[msg.sender]);
195
196     // SafeMath.sub will throw if there is not enough balance.
197     balances[msg.sender] = balances[msg.sender].sub(_value);
198     balances[_to] = balances[_to].add(_value);
199     emit Transfer(msg.sender, _to, _value);
200     return true;
201   }
202
203   /**
204   * @dev Gets the balance of the specified address.
205   * @param _owner The address to query the the balance of.
206   * @return An uint256 representing the amount owned by the passed address.
207   */
208   function balanceOf(address _owner) public view returns (uint256 balance) {
209     return balances[_owner];
210   }
211
212 }
213
214 contract StandardToken is ERC20, BasicToken {
215   mapping (address => mapping (address => uint256)) internal allowed;
216
217   /**
218    * @dev Transfer tokens from one address to another
219    * @param _from address The address which you want to send tokens from
220    * @param _to address The address which you want to transfer to
221    * @param _value uint256 the amount of tokens to be transferred
222    */
223   /*@CTK "transferFrom correctness"
224     @tag assume_completion
225     @post _to != address(0)
```

```
226     @post _value <= balances[_from]
227     @post _value <= allowed[_from][msg.sender]
228     @post _from != _to -> __post.balances[_from] == balances[_from] - _value
229     @post _from != _to -> __post.balances[_to] == balances[_to] + _value
230     @post _from == _to -> __post.balances[_from] == balances[_from]
231     @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
232     @post !(__has_overflow)
233     @post !(__has_buf_overflow)
234     @post !(__has_assertion_failure)
235     */
236   function transferFrom(address _from, address _to, uint256 _value) public returns (
          bool) {
237     require(_to != address(0));
238     require(_value <= balances[_from]);
239     require(_value <= allowed[_from][msg.sender]);
240
241     balances[_from] = balances[_from].sub(_value);
242     balances[_to] = balances[_to].add(_value);
243     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
244     emit Transfer(_from, _to, _value);
245     return true;
246   }
247
248   /**
249    * @dev Approve the passed address to spend the specified amount of tokens on behalf
           of msg.sender.
250    * @param _spender The address which will spend the funds.
251    * @param _value The amount of tokens to be spent.
252    */
253   //@CTK NO_OVERFLOW
254   //@CTK NO_BUF_OVERFLOW
255   //@CTK NO_ASF
256   /*@CTK "approve correctness"
257     @post __post.allowed[msg.sender][_spender] == _value
258     @post __return
259     */
260   function approve(address _spender, uint256 _value) public returns (bool) {
261     allowed[msg.sender][_spender] = _value;
262     emit Approval(msg.sender, _spender, _value);
263     return true;
264   }
265
266   /**
267    * @dev Function to check the amount of tokens that an owner allowed to a spender.
268    * @param _owner address The address which owns the funds.
269    * @param _spender address The address which will spend the funds.
270    * @return A uint256 specifying the amount of tokens still available for the spender
           .
271    */
272   /*@CTK allowance
273     @post __return == allowed[_owner][_spender]
274    */
275   function allowance(address _owner, address _spender) public view returns (uint256) {
276     return allowed[_owner][_spender];
277   }
278
279   /**
280    * @dev Increase the amount of tokens that an owner allowed to a spender.
```

```
281     * approve should be called when allowed[_spender] == 0. To increment
282     * allowed value is better to use this function to avoid 2 calls (and wait until
283     * the first transaction is mined)
284     * @param _spender The address which will spend the funds.
285     * @param _addedValue The amount of tokens to increase the allowance by.
286     */
287    /*@CTK increaseApproval
288      @tag assume_completion
289      @post __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] +
           _addedValue
290     */
291    function increaseApproval(address _spender, uint _addedValue) public returns (bool)
         {
292      allowed[msg.sender][_spender] = allowed[msg.sender][_spender].add(_addedValue);
293      emit  Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
294      return true;
295    }
296
297    /**
298     * @dev Decrease the amount of tokens that an owner allowed to a spender.
299     * approve should be called when allowed[_spender] == 0. To decrement
300     * allowed value is better to use this function to avoid 2 calls (and wait until
301     * the first transaction is mined)
302     * @param _spender The address which will spend the funds.
303     * @param _subtractedValue The amount of tokens to decrease the allowance by.
304     */
305    /*@CTK decreaseApproval
306      @tag assume_completion
307      @post (_subtractedValue > allowed[msg.sender][_spender]) -> __post.allowed[msg.
           sender][_spender] == 0
308      @post (_subtractedValue <= allowed[msg.sender][_spender]) -> __post.allowed[msg.
           sender][_spender] == allowed[msg.sender][_spender] - _subtractedValue
309     */
310    function decreaseApproval(address _spender, uint _subtractedValue) public returns (
         bool) {
311      uint oldValue = allowed[msg.sender][_spender];
312      if (_subtractedValue > oldValue) {
313        allowed[msg.sender][_spender] = 0;
314      } else {
315        allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
316      }
317    emit  Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
318      return true;
319    }
320
321  }
322
323  contract PausableToken is StandardToken, Pausable {
324
325    function transfer(address _to, uint256 _value) public whenNotPaused returns (bool) {
326      return super.transfer(_to, _value);
327    }
328
329    function transferFrom(address _from, address _to, uint256 _value) public
         whenNotPaused returns (bool) {
330      return super.transferFrom(_from, _to, _value);
331    }
332
```

```
333   function approve(address _spender, uint256 _value) public whenNotPaused returns (
          bool) {
334     return super.approve(_spender, _value);
335   }
336
337   function increaseApproval(address _spender, uint _addedValue) public whenNotPaused
          returns (bool success) {
338     return super.increaseApproval(_spender, _addedValue);
339   }
340
341   function decreaseApproval(address _spender, uint _subtractedValue) public
          whenNotPaused returns (bool success) {
342     return super.decreaseApproval(_spender, _subtractedValue);
343   }
344 }
345
346 /**
347  * @dev Initialize contract basic information
348  */
349 contract OneDSToken is PausableToken {
350     string public name = "OneDS";
351     string public symbol = "ONE";
352     uint public decimals = 18;
353     uint public INITIAL_SUPPLY = 10000000000000000000000000;
354
355     /*@CTK OneDSToken
356       @tag assume_completion
357       @post __post.balances[msg.sender] == __post.totalSupply
358       @post !(__has_overflow)
359       @post !(__has_buf_overflow)
360       @post !(__has_assertion_failure)
361      */
362     function OneDSToken() public {
363         totalSupply = INITIAL_SUPPLY;
364         balances[msg.sender] = INITIAL_SUPPLY;
365     }
366 }
```

File Migrations.sol

```
1   pragma solidity ^0.4.17;
2
3   contract Migrations {
4     address public owner;
5     uint public last_completed_migration;
6
7     modifier restricted() {
8       if (msg.sender == owner) _;
9     }
10
11    //@CTK NO_OVERFLOW
12    //@CTK NO_BUF_OVERFLOW
13    //@CTK NO_ASF
14    /*@CTK "restricted"
15      @post (__post.owner) == (msg.sender)
16    */
17    /*@CTK "migrations"
18      @post (__post.owner) == (msg.sender)
19    */
```

```
20    /*@CTK "init_migrations"
21      @post (__post.owner) == (msg.sender)
22    */
23    /*@CTK "Migrations"
24      @post (__post.owner) == (msg.sender)
25    */
26    function Migrations() public {
27      owner = msg.sender;
28    }
29
30    //@CTK NO_OVERFLOW
31    //@CTK NO_BUF_OVERFLOW
32    //@CTK NO_ASF
33    /*@CTK "set_complete"
34      @pre (msg.sender) == (owner)
35      @post (__post.last_completed_migration) == (completed)
36    */
37    /*@CTK "setCompleted"
38      @pre (msg.sender) == (owner)
39      @post (__post.last_completed_migration) == (completed)
40    */
41    /*@CTK "setCompleted"
42      @post ((__post.last_completed_migration) != (completed)) -> ((msg.sender) != (
          owner))
43    */
44    function setCompleted(uint completed) public restricted {
45      last_completed_migration = completed;
46    }
47
48    //@CTK NO_OVERFLOW
49    //@CTK NO_BUF_OVERFLOW
50    //@CTK NO_ASF
51    function upgrade(address new_address) public restricted {
52      Migrations upgraded = Migrations(new_address);
53      upgraded.setCompleted(last_completed_migration);
54    }
55  }
```