CertiK Verification Report For NKN

Request Date: 2018-12-23
Revision Date: 2018-12-27

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 NKN(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.

PASS

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

Dec 27, 2018

Summary

This is the report for smart contract verification service requestd by NKN. The goal of the audition is to guarantee that verified smart contracts are robust enough to avoid potentially unexpected loopholes.
The result of this report is only a reflection of the source code that was determined in this scope, and of the source code at the audit time.

Type of Issues

CertiK smart label engine applied 100% coveraged formal verification labels on the source code, and scanned the code by 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Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0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

Allowance is mutated even when sender and receiver are the same.

In function `transferFrom`, allowance of the sender is mutated even in the case when sender and receiver are the same. This is only a small issue that author can consider handling.

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

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

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

Source Code with CertiK Labels

File NKNToken.sol

```
pragma solidity ^0.4.24;

/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
library SafeMath {
    /*@CTK "SafeMath mul"
        @post (a > 0) && (((a * b) / a) != b) -> __reverted
        @post __reverted -> (a > 0) && (((a * b) / a) != b)
        @post !__reverted -> __return == a * b
        @post !__reverted == !__has_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 "SafeMath div"
        @post b != 0 -> !__reverted
        @post !__reverted -> __return == a / b
        @post !__reverted -> !__has_overflow
    */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b > 0);
        uint256 c = a / b;
        return c;
    }

    /*@CTK "SafeMath sub"
        @post (a < b) == __reverted
        @post !__reverted -> __return == a - b
        @post !__reverted -> !__has_overflow
    */
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }

    /*@CTK "SafeMath add"
        @post (a + b < a || a + b < b) == __reverted
        @post !__reverted -> __return == a + b
        @post !__reverted -> !__has_overflow
    */
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        assert(c >= a);
        return c;
    }
}
```

```
/**
 * @title ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 */
contract ERC20 {
    uint256 public totalSupply;
    function balanceOf(address who) public view returns (uint256);
    function transfer(address to, uint256 value) public returns (bool);
    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 Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

contract Owned {
    address public owner;

    event OwnershipTransferred(address indexed _from, address indexed _to);

    /*@CTK Owned
        @post __post.owner == msg.sender
     */
    constructor() public {
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    /*@CTK transferOwnership
        @tag assume_completion
        @post msg.sender == owner
        @post address(0) != _owner
        @post __post.owner == _owner
    */
    function transferOwnership(address _owner) onlyOwner public {
        require(_owner != address(0));
        owner = _owner;

        emit OwnershipTransferred(owner, _owner);
    }
}

contract ERC20Token is ERC20, Owned {
    using SafeMath for uint256;

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


    // True if transfers are allowed
    bool public transferable = false;
```

```
    modifier canTransfer() {
        require(transferable == true);
        _;
    }

    /*@CTK setTransferable
        @tag assume_completion
        @post msg.sender == owner
        @post __post.transferable == _transferable
     */
    function setTransferable(bool _transferable) onlyOwner public {
        transferable = _transferable;
    }

    /**
     * @dev transfer token for a specified address
     * @param _to The address to transfer to.
     * @param _value The amount to be transferred.
     */
    /*@CTK transfer
        @tag assume_completion
        @pre msg.sender != _to
        @post transferable == true
        @post __post.balances[msg.sender] == balances[msg.sender] - _value
        @post __post.balances[_to] == balances[_to] + _value
     */
    function transfer(address _to, uint256 _value) canTransfer 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 balanceOf
        @post balance == balances[_owner]
    */
    function balanceOf(address _owner) public view returns (uint256 balance) {
        return balances[_owner];
    }

    /**
    * @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 transferFrom
        @tag assume_completion
        @pre _from != _to
```

```
        @post __post.balances[_from] == balances[_from] - _value
        @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
        @post __post.balances[_to] == balances[_to] + _value
    */
    /*@CTK FAIL "transferFrom_sameAddress"
        @tag assume_completion
        @pre _from == _to
        @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender]
    */
    function transferFrom(address _from, address _to, uint256 _value) canTransfer
        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;
    }

    // Allow `_spender` to withdraw from your account, multiple times.
    /*@CTK approve
        @tag assume_completion
        @post (allowed[msg.sender][_spender] == 0) || (_value == 0)
        @post __post.allowed[msg.sender][_spender] == _value
     */
    function approve(address _spender, uint _value) public returns (bool success) {
        // To change the approve amount you first have to reduce the addresses`
        //  allowance to zero by calling `approve(_spender, 0)` if it is not
        //  already 0 to mitigate the race condition described here:
        //  https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
        if ((_value != 0) && (allowed[msg.sender][_spender] != 0)) {
            assert(false);
            // revert();
        }
        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 allowance
        @post __return == allowed[_owner][_spender]
    */
    function allowance(address _owner, address _spender) public view returns (uint256)
         {
        return allowed[_owner][_spender];
    }

    function () public payable {
```

```
        revert();
    }
}

contract NKNToken is ERC20Token{
    string public name = "NKN";
    string public symbol = "NKN";
    uint8 public decimals = 18;

    uint256 public totalSupplyCap = 7 * 10**8 * 10**uint256(decimals);

    constructor(address _issuer) public {
        totalSupply = totalSupplyCap;
        balances[_issuer] = totalSupplyCap;
        emit Transfer(address(0), _issuer, totalSupplyCap);
    }
}
```

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
```

Static Analysis Request

INSECURE_COMPILER_VERSION

Line 1 in File NKNToken.sol

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

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

Formal Verification Request 1

SafeMath mul

27, Dec 2018
483.48 ms

Line 8-13 in File NKNToken.sol

```
8   /*@CTK "SafeMath mul"
9       @post (a > 0) && (((a * b) / a) != b) -> __reverted
10      @post __reverted -> (a > 0) && (((a * b) / a) != b)
11      @post !__reverted -> __return == a * b
12      @post !__reverted == !__has_overflow
13  */
```

Line 14-21 in File NKNToken.sol

```
14  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
15      if (a == 0) {
16          return 0;
17      }
18      uint256 c = a * b;
19      assert(c / a == b);
20      return c;
21  }
```

The code meets the specification

Formal Verification Request 2

SafeMath div

27, Dec 2018
15.19 ms

Line 23-27 in File NKNToken.sol

```
23  /*@CTK "SafeMath div"
24      @post b != 0 -> !__reverted
25      @post !__reverted -> __return == a / b
26      @post !__reverted -> !__has_overflow
27  */
```

Line 28-32 in File NKNToken.sol

```
28  function div(uint256 a, uint256 b) internal pure returns (uint256) {
29      assert(b > 0);
30      uint256 c = a / b;
31      return c;
32  }
```

The code meets the specification

Formal Verification Request 3

SafeMath sub

27, Dec 2018
13.37 ms

Line 34-38 in File NKNToken.sol

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

Line 39-42 in File NKNToken.sol

```
39  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
40      assert(b <= a);
41      return a - b;
42  }
```

The code meets the specification

Formal Verification Request 4

SafeMath add

27, Dec 2018
16.76 ms

Line 44-48 in File NKNToken.sol

```
44  /*@CTK "SafeMath add"
45      @post (a + b < a || a + b < b) == __reverted
46      @post !__reverted -> __return == a + b
47      @post !__reverted -> !__has_overflow
48  */
```

Line 49-53 in File NKNToken.sol

```
49  function add(uint256 a, uint256 b) internal pure returns (uint256) {
50      uint256 c = a + b;
51      assert(c >= a);
52      return c;
53  }
```

The code meets the specification

Formal Verification Request 5

Owned

27, Dec 2018
5.37 ms

Line 77-79 in File NKNToken.sol

```
77  /*@CTK Owned
78      @post __post.owner == msg.sender
79   */
```

Line 80-82 in File NKNToken.sol

```
80  constructor() public {
81      owner = msg.sender;
82  }
```

The code meets the specification

Formal Verification Request 6

transferOwnership

27, Dec 2018
24.02 ms

Line 89-94 in File NKNToken.sol

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

Line 95-100 in File NKNToken.sol

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

The code meets the specification

Formal Verification Request 7

setTransferable

27, Dec 2018
17.52 ms

Line 118-122 in File NKNToken.sol

```
118 /*@CTK setTransferable
119     @tag assume_completion
120     @post msg.sender == owner
121     @post __post.transferable == _transferable
122  */
```

Line 123-125 in File NKNToken.sol

```
123 function setTransferable(bool _transferable) onlyOwner public {
124     transferable = _transferable;
125 }
```

The code meets the specification

Formal Verification Request 8

transfer

27, Dec 2018
231.91 ms

Line 132-138 in File NKNToken.sol

```
132 /*@CTK transfer
133     @tag assume_completion
134     @pre msg.sender != _to
135     @post transferable == true
136     @post __post.balances[msg.sender] == balances[msg.sender] - _value
137     @post __post.balances[_to] == balances[_to] + _value
138  */
```

Line 139-147 in File NKNToken.sol

```
139 function transfer(address _to, uint256 _value) canTransfer public returns (bool) {
140     require(_to != address(0));
141     require(_value <= balances[msg.sender]);
142
143     balances[msg.sender] = balances[msg.sender].sub(_value);
144     balances[_to] = balances[_to].add(_value);
145     emit Transfer(msg.sender, _to, _value);
146     return true;
147 }
```

The code meets the specification

Formal Verification Request 9

balanceOf

27, Dec 2018
5.92 ms

Line 154-156 in File NKNToken.sol

```
154 /*@CTK balanceOf
155     @post balance == balances[_owner]
156 */
```

Line 157-159 in File NKNToken.sol

```
157 function balanceOf(address _owner) public view returns (uint256 balance) {
158     return balances[_owner];
159 }
```

The code meets the specification

Formal Verification Request 10

transferFrom

27, Dec 2018
460.17 ms

Line 167-173 in File NKNToken.sol

```
/*@CTK transferFrom
    @tag assume_completion
    @pre _from != _to
    @post __post.balances[_from] == balances[_from] - _value
    @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender] - _value
    @post __post.balances[_to] == balances[_to] + _value
*/
```

Line 179-189 in File NKNToken.sol

```
function transferFrom(address _from, address _to, uint256 _value) canTransfer
    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 11

transferFrom_sameAddress

27, Dec 2018
97.08 ms

Line 174-178 in File NKNToken.sol

```
/*@CTK FAIL "transferFrom_sameAddress"
    @tag assume_completion
    @pre _from == _to
    @post __post.allowed[_from][msg.sender] == allowed[_from][msg.sender]
*/
```

Line 179-189 in File NKNToken.sol

```
    function transferFrom(address _from, address _to, uint256 _value) canTransfer
        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;
    }
```

This code violates the specification

```
Counter Example:
Before Execution:
    Input = {
        _from = 128
        _to = 128
        _value = 28
    }
    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": "ERC20Token",
          "balance": 0,
          "contract": {
            "balances": [
              {
                "key": 128,
                "value": 64
              },
              {
                "key": 16,
                "value": 1
              },
              {
```

```
              "key": "ALL_OTHERS",
              "value": 36
            }
          ],
          "allowed": [
            {
              "key": 128,
              "value": [
                {
                  "key": 0,
                  "value": 32
                },
                {
                  "key": 16,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 36
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 36
                }
              ]
            }
          ],
          "transferable": true,
          "owner": 0,
          "totalSupply": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

After Execution:
    Input = {
        _from = 128
        _to = 128
        _value = 28
    }
    This = 0
    Internal = {
        __has_assertion_failure = false
        __has_buf_overflow = false
        __has_overflow = false
        __has_returned = true
        __reverted = false
        msg = {
```

```
        "gas": 0,
        "sender": 0,
        "value": 0
      }
  }
  Other = {
      __return = true
      block = {
        "number": 0,
        "timestamp": 0
      }
  }
  Address_Map = [
    {
      "key": 0,
      "value": {
        "contract_name": "ERC20Token",
        "balance": 0,
        "contract": {
          "balances": [
            {
              "key": 128,
              "value": 64
            },
            {
              "key": 16,
              "value": 1
            },
            {
              "key": "ALL_OTHERS",
              "value": 36
            }
          ],
          "allowed": [
            {
              "key": 128,
              "value": [
                {
                  "key": 0,
                  "value": 4
                },
                {
                  "key": 16,
                  "value": 0
                },
                {
                  "key": "ALL_OTHERS",
                  "value": 36
                }
              ]
            },
            {
              "key": "ALL_OTHERS",
              "value": [
                {
                  "key": "ALL_OTHERS",
                  "value": 36
                }
```

```
              ]
            }
          ],
          "transferable": true,
          "owner": 0,
          "totalSupply": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]
```

Formal Verification Request 12

approve

📅 27, Dec 2018
⏱ 30.34 ms

Line 192-196 in File NKNToken.sol

```
/*@CTK approve
    @tag assume_completion
    @post (allowed[msg.sender][_spender] == 0) || (_value == 0)
    @post __post.allowed[msg.sender][_spender] == _value
 */
```

Line 197-209 in File NKNToken.sol

```
function approve(address _spender, uint _value) public returns (bool success) {
    // To change the approve amount you first have to reduce the addresses`
    //  allowance to zero by calling `approve(_spender, 0)` if it is not
    //  already 0 to mitigate the race condition described here:
    //  https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
    if ((_value != 0) && (allowed[msg.sender][_spender] != 0)) {
        assert(false);
        // revert();
    }
    allowed[msg.sender][_spender] = _value;
    emit Approval(msg.sender, _spender, _value);
    return true;
}
```

✅ The code meets the specification

Formal Verification Request 13

allowance

📅 27, Dec 2018
⏱ 6.39 ms

Line 217-219 in File NKNToken.sol

```
217     /*@CTK allowance
218         @post __return == allowed[_owner][_spender]
219     */
```

Line 220-222 in File NKNToken.sol

```
220     function allowance(address _owner, address _spender) public view returns (uint256)
            {
221         return allowed[_owner][_spender];
222     }
```

✅ The code meets the specification