CertiK Audit Report For BenepitToken

Request Date: 2019-06-12
Revision Date: 2019-06-20
Platform Name: Ethereum

Contents

Disclaimer

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 Benepit-Token. This audit was conducted to discover issues and vulnerabilities in the source code of BenepitToken's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis and Manual Review, a comprehensive examination has been performed. The auditing process pays special attention to the following considerations.

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

Vulnerability Classification

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

Critical

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

Medium

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

Low

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

Testing Summary

PASS

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

Jun 20, 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.

(Note: Violations in formal verification result section are for internal evaluation only and are not indication of security issue in the client code.)

Manual Review Notes

Review Details

Source Code SHA-256 Checksum

- **BenepitToken.sol**
  `9cfe778cc77ca1f1cb8c86b6513c455b8df8c0006fb0e84e431332a679891653`

Summary

CertiK was chosen by Benepit to audit the design and implementation of its BenepitToken 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.

Recommendations

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

Ownable

- **transferOwnership()** — Recommend using pull model instead of push model when changing the owner of the contract to further reduce the risk manual error.

```
address owner;
address proposedOwner;
function proposeNewOwner(address newOwner) isOwner public {
    require(newOwner != address(0), ...);
    require(newOwner != address(0), ...);
    proposedOwner = newOwner;
    // emit LogOwnerTransferProposed ...
}
function claimOwnership() public {
    require(msg.sender == proposedOwner, ...);
    owner = proposedOwner;
    proposedOwner = address(0);
    // emit LogOwnerTransferred ...
}
```

Static Analysis Results

INSECURE_COMPILER_VERSION

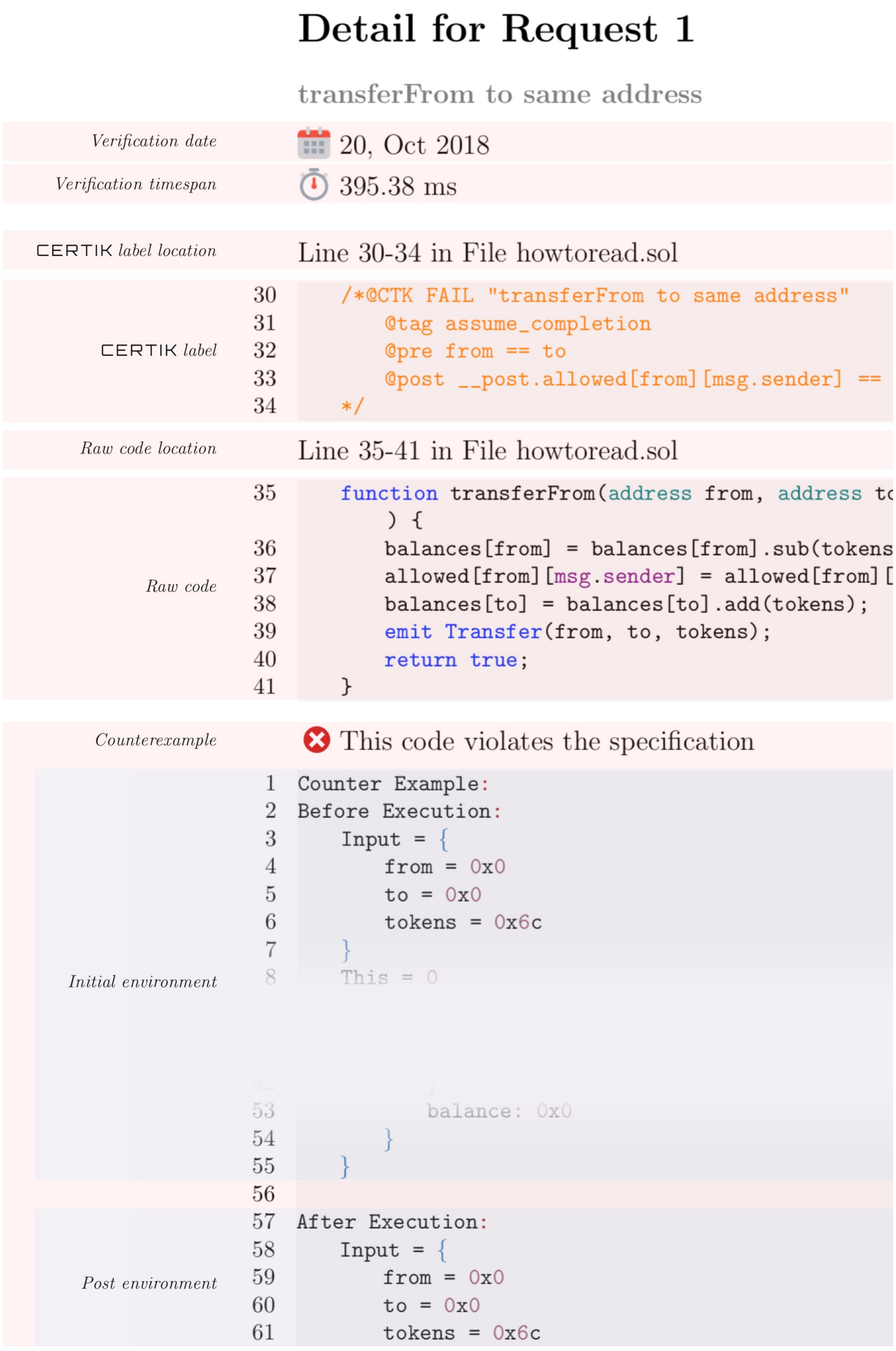
Line 1 in File BenepitToken.sol

```
pragma solidity 0.4.24;
```

❗ Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor_0.4.x, IncorrectEventSignatureInLibraries_0.4.x, ABIEncoderV2PackedStorage_0.4.x, ExpExponentCleanup, EventStructWrongData

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

```
Counter Example:
Before Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
    }
    This = 0

            balance: 0x0
        }
    }

```

Post environment

```
After Execution:
    Input = {
        from = 0x0
        to = 0x0
        tokens = 0x6c
```

Formal Verification Request 1

Method will not encounter an assertion failure.

20, Jun 2019
19.94 ms

Line 26 in File BenepitToken.sol

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

Line 34-44 in File BenepitToken.sol

```
function mul(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
  // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
  // benefit is lost if 'b' is also tested.
  if (_a == 0) {
    return 0;
  }

  c = _a * _b;
  assert(c / _a == _b);
  return c;
}
```

❌ This code violates the specification.

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

Function invocation is reverted.
```

Formal Verification Request 2

SafeMath mul

📅 20, Jun 2019
⏱ 306.19 ms

Line 27-33 in File BenepitToken.sol

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

Line 34-44 in File BenepitToken.sol

```
function mul(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
  // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
  // benefit is lost if 'b' is also tested.
  if (_a == 0) {
    return 0;
  }

  c = _a * _b;
  assert(c / _a == _b);
  return c;
}
```

✓ The code meets the specification.

Formal Verification Request 3

Method will not encounter an assertion failure.

📅 20, Jun 2019
⏱ 5.41 ms

Line 49 in File BenepitToken.sol

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

Line 57-62 in File BenepitToken.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 4

SafeMath div

📅 20, Jun 2019
⏱ 0.32 ms

Line 50-56 in File BenepitToken.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 57-62 in File BenepitToken.sol

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

✅ The code meets the specification.

Formal Verification Request 5

Method will not encounter an assertion failure.

📅 20, Jun 2019
⏱ 10.69 ms

Line 67 in File BenepitToken.sol

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

Line 75-78 in File BenepitToken.sol

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

❌ This code violates the specification.

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

Function invocation is reverted.
```

Formal Verification Request 6

SafeMath sub

📅 20, Jun 2019

⏱ 0.84 ms

Line 68-74 in File BenepitToken.sol

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

Line 75-78 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 7

Method will not encounter an assertion failure.

📅 20, Jun 2019
⏱ 11.26 ms

Line 83 in File BenepitToken.sol

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

Line 91-95 in File BenepitToken.sol

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

✗ This code violates the specification.

```
Counter Example:
Before Execution:
    Input = {
        _a = 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 8

SafeMath add

📅 20, Jun 2019
⏱ 2.56 ms

Line 84-90 in File BenepitToken.sol

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

Line 91-95 in File BenepitToken.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 9

If method completes, integer overflow would not happen.

📅 20, Jun 2019
⏱ 5.34 ms

Line 114 in File BenepitToken.sol

```
  //@CTK NO_OVERFLOW
```

Line 121-123 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 10

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

20, Jun 2019
0.39 ms

Line 115 in File BenepitToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 121-123 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 11

Method will not encounter an assertion failure.

20, Jun 2019
0.35 ms

Line 116 in File BenepitToken.sol

```
//@CTK NO_ASF
```

Line 121-123 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 12

totalSupply

20, Jun 2019
0.32 ms

Line 117-120 in File BenepitToken.sol

```
/*@CTK totalSupply
  @tag assume_completion
  @post (__return) == (totalSupply_)
*/
```

Line 121-123 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 13

If method completes, integer overflow would not happen.

20, Jun 2019
79.35 ms

Line 130 in File BenepitToken.sol

```
//@CTK NO_OVERFLOW
```

Line 143-151 in File BenepitToken.sol

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

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

✓ The code meets the specification.

Formal Verification Request 14

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

20, Jun 2019
10.48 ms

Line 131 in File BenepitToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 143-151 in File BenepitToken.sol

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

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

✓ The code meets the specification.

Formal Verification Request 15

Method will not encounter an assertion failure.

📅 20, Jun 2019
⏱ 42.72 ms

Line 132 in File BenepitToken.sol

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

Line 143-151 in File BenepitToken.sol

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

  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 = 4
    }
    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": 4,
                "value": 252
              },
              {
                "key": 0,
                "value": 4
              },
              {
                "key": "ALL_OTHERS",
                "value": 255
              }
            ],
            "totalSupply_": 0
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 16

transfer

📅 20, Jun 2019
⏱ 149.97 ms

Line 133-142 in File BenepitToken.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 143-151 in File BenepitToken.sol

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

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

✓ The code meets the specification.

Formal Verification Request 17

If method completes, integer overflow would not happen.

📅 20, Jun 2019
⏱ 4.44 ms

Line 158 in File BenepitToken.sol

```
//@CTK NO_OVERFLOW
```

Line 165-167 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 18

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

📅 20, Jun 2019
⏱ 0.5 ms

Line 159 in File BenepitToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 165-167 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 19

Method will not encounter an assertion failure.

📅 20, Jun 2019
⏱ 0.32 ms

Line 160 in File BenepitToken.sol

```
//@CTK NO_ASF
```

Line 165-167 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 20

balanceOf

📅 20, Jun 2019
⏱ 0.34 ms

Line 161-164 in File BenepitToken.sol

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

Line 165-167 in File BenepitToken.sol

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

✅ The code meets the specification.

Formal Verification Request 21

If method completes, integer overflow would not happen.

📅 20, Jun 2019
⏱ 100.36 ms

Line 212 in File BenepitToken.sol

```
//@CTK NO_OVERFLOW
```

Line 227-244 in File BenepitToken.sol

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

  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 22

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

📅 20, Jun 2019
⏱ 10.56 ms

Line 213 in File BenepitToken.sol

```
213  //@CTK NO_BUF_OVERFLOW
```

Line 227-244 in File BenepitToken.sol

```
227  function transferFrom(
228    address _from,
229    address _to,
230    uint256 _value
231  )
232    public
233    returns (bool)
234  {
235    require(_value <= balances[_from]);
236    require(_value <= allowed[_from][msg.sender]);
237    require(_to != address(0));
238
239    balances[_from] = balances[_from].sub(_value);
240    balances[_to] = balances[_to].add(_value);
241    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
242    emit Transfer(_from, _to, _value);
243    return true;
244  }
```

✅ The code meets the specification.

Formal Verification Request 23

Method will not encounter an assertion failure.

📅 20, Jun 2019
⏱ 63.24 ms

Line 214 in File BenepitToken.sol

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

Line 227-244 in File BenepitToken.sol

```
227  function transferFrom(
228    address _from,
229    address _to,
230    uint256 _value
231  )
232    public
233    returns (bool)
234  {
235    require(_value <= balances[_from]);
236    require(_value <= allowed[_from][msg.sender]);
237    require(_to != address(0));
238
```

```
    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 = 0
        _to = 4
        _value = 137
    }
    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": 128,
                "value": [
                  {
                    "key": 0,
                    "value": 0
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 137
                  }
                ]
              },
              {
                "key": 0,
                "value": [
                  {
                    "key": 0,
                    "value": 200
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 137
                  }
                ]
              },
              {
                "key": 64,
                "value": [
                  {
                    "key": 0,
                    "value": 0
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 137
                  }
                ]
              },
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 119
                  }
                ]
              }
            ],
            "balances": [
              {
                "key": 32,
                "value": 0
              },
              {
                "key": 128,
                "value": 0
              },
              {
                "key": 64,
                "value": 32
              },
              {
                "key": 0,
                "value": 164
              },
              {
                "key": 68,
                "value": 0
              },
              {
                "key": 4,
                "value": 121
            },
            {
              "key": 8,
              "value": 4
            },
            {
              "key": "ALL_OTHERS",
              "value": 119
            }
          ],
          "totalSupply_": 0
        }
      }
    },
    {
      "key": "ALL_OTHERS",
      "value": "EmptyAddress"
    }
  ]

Function invocation is reverted.
```

Formal Verification Request 24

transferFrom

📅 20, Jun 2019
⏱ 330.71 ms

Line 215-226 in File BenepitToken.sol

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

Line 227-244 in File BenepitToken.sol

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

```

```
239     balances[_from] = balances[_from].sub(_value);
240     balances[_to] = balances[_to].add(_value);
241     allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
242     emit Transfer(_from, _to, _value);
243     return true;
244   }
```

The code meets the specification.

Formal Verification Request 25

If method completes, integer overflow would not happen.

20, Jun 2019
7.3 ms

Line 255 in File BenepitToken.sol

```
255   //@CTK NO_OVERFLOW
```

Line 262-266 in File BenepitToken.sol

```
262   function approve(address _spender, uint256 _value) public returns (bool) {
263     allowed[msg.sender][_spender] = _value;
264     emit Approval(msg.sender, _spender, _value);
265     return true;
266   }
```

The code meets the specification.

Formal Verification Request 26

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

20, Jun 2019
0.32 ms

Line 256 in File BenepitToken.sol

```
256   //@CTK NO_BUF_OVERFLOW
```

Line 262-266 in File BenepitToken.sol

```
262   function approve(address _spender, uint256 _value) public returns (bool) {
263     allowed[msg.sender][_spender] = _value;
264     emit Approval(msg.sender, _spender, _value);
265     return true;
266   }
```

The code meets the specification.

Formal Verification Request 27

Method will not encounter an assertion failure.

📅 20, Jun 2019
⏱ 0.3 ms

Line 257 in File BenepitToken.sol

```
//@CTK NO_ASF
```

Line 262-266 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 28

approve

📅 20, Jun 2019
⏱ 1.13 ms

Line 258-261 in File BenepitToken.sol

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

Line 262-266 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 29

If method completes, integer overflow would not happen.

📅 20, Jun 2019
⏱ 4.81 ms

Line 274 in File BenepitToken.sol

```
//@CTK NO_OVERFLOW
```

Line 281-290 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 30

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

📅 20, Jun 2019
⏱ 0.3 ms

Line 275 in File BenepitToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 281-290 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 31

Method will not encounter an assertion failure.

📅 20, Jun 2019
⏱ 0.3 ms

Line 276 in File BenepitToken.sol

```
//@CTK NO_ASF
```

Line 281-290 in File BenepitToken.sol

```
function allowance(
  address _owner,
  address _spender
 )
  public
```

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

✓ The code meets the specification.

Formal Verification Request 32

allowance

20, Jun 2019
0.3 ms

Line 277-280 in File BenepitToken.sol

```
  /*@CTK allowance
    @tag assume_completion
    @post (__return) == (allowed[_owner][_spender])
  */
```

Line 281-290 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 33

If method completes, integer overflow would not happen.

20, Jun 2019
34.08 ms

Line 301 in File BenepitToken.sol

```
  //@CTK NO_OVERFLOW
```

Line 311-322 in File BenepitToken.sol

```
  function increaseApproval(
    address _spender,
    uint256 _addedValue
  )
    public
    returns (bool)
  {
```

```
318     allowed[msg.sender][_spender] = (
319       allowed[msg.sender][_spender].add(_addedValue));
320     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
321     return true;
322   }
```

✅ The code meets the specification.

Formal Verification Request 34

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

📅 20, Jun 2019
⏱ 0.75 ms

Line 302 in File BenepitToken.sol

```
302   //@CTK NO_BUF_OVERFLOW
```

Line 311-322 in File BenepitToken.sol

```
311   function increaseApproval(
312     address _spender,
313     uint256 _addedValue
314   )
315     public
316     returns (bool)
317   {
318     allowed[msg.sender][_spender] = (
319       allowed[msg.sender][_spender].add(_addedValue));
320     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
321     return true;
322   }
```

✅ The code meets the specification.

Formal Verification Request 35

Method will not encounter an assertion failure.

📅 20, Jun 2019
⏱ 7.31 ms

Line 303 in File BenepitToken.sol

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

Line 311-322 in File BenepitToken.sol

```
311   function increaseApproval(
312     address _spender,
313     uint256 _addedValue
314   )
315     public
316     returns (bool)
317   {
318     allowed[msg.sender][_spender] = (
    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 = 161
        _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": 95
                  },
                  {
                    "key": 4,
                    "value": 128
                  },
                  {
                    "key": "ALL_OTHERS",
                    "value": 161
                  }
                ]
              },
              {
                "key": "ALL_OTHERS",
                "value": [
                  {
                    "key": "ALL_OTHERS",
                    "value": 161
                  }
                ]
              }
            ],
            "balances": [
              {
                "key": 4,
                "value": 0
              },
              {
                "key": "ALL_OTHERS",
                "value": 161
              }
            ],
            "totalSupply_": 0
          }
        }
      },
      {
        "key": "ALL_OTHERS",
        "value": "EmptyAddress"
      }
    ]

Function invocation is reverted.
```

Formal Verification Request 36

increaseApproval

📅 20, Jun 2019
⏱ 2.17 ms

Line 304-310 in File BenepitToken.sol

```
  /*@CTK increaseApproval
    @tag assume_completion
    @pre _spender != 0x0
    @pre _spender != msg.sender
    @post (__post.allowed[msg.sender][_spender]) == (allowed[msg.sender][_spender] +
        _addedValue)
    @post (__return) == (true)
  */
```

Line 311-322 in File BenepitToken.sol

```
  function increaseApproval(
    address _spender,
    uint256 _addedValue
  )
    public
    returns (bool)
  {
```

```
318     allowed[msg.sender][_spender] = (
319       allowed[msg.sender][_spender].add(_addedValue));
320     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
321     return true;
322   }
```

✅ The code meets the specification.

Formal Verification Request 37

If method completes, integer overflow would not happen.

📅 20, Jun 2019
⏱ 43.37 ms

Line 333 in File BenepitToken.sol

```
333   //@CTK NO_OVERFLOW
```

Line 343-358 in File BenepitToken.sol

```
343   function decreaseApproval(
344     address _spender,
345     uint256 _subtractedValue
346   )
347     public
348     returns (bool)
349   {
350     uint256 oldValue = allowed[msg.sender][_spender];
351     if (_subtractedValue >= oldValue) {
352       allowed[msg.sender][_spender] = 0;
353     } else {
354       allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
355     }
356     emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
357     return true;
358   }
```

✅ The code meets the specification.

Formal Verification Request 38

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

📅 20, Jun 2019
⏱ 0.89 ms

Line 334 in File BenepitToken.sol

```
334   //@CTK NO_BUF_OVERFLOW
```

Line 343-358 in File BenepitToken.sol

```
343   function decreaseApproval(
344     address _spender,
345     uint256 _subtractedValue
346   )
```

```
  public
  returns (bool)
{
  uint256 oldValue = allowed[msg.sender][_spender];
  if (_subtractedValue >= oldValue) {
    allowed[msg.sender][_spender] = 0;
  } else {
    allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
  }
  emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
  return true;
}
```

✓ The code meets the specification.

Formal Verification Request 39

Method will not encounter an assertion failure.

20, Jun 2019
5.26 ms

Line 335 in File BenepitToken.sol

```
//@CTK NO_ASF
```

Line 343-358 in File BenepitToken.sol

```
function decreaseApproval(
  address _spender,
  uint256 _subtractedValue
)
  public
  returns (bool)
{
  uint256 oldValue = allowed[msg.sender][_spender];
  if (_subtractedValue >= oldValue) {
    allowed[msg.sender][_spender] = 0;
  } else {
    allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
  }
  emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
  return true;
}
```

✓ The code meets the specification.

Formal Verification Request 40

decreaseApproval

20, Jun 2019
28.37 ms

Line 336-342 in File BenepitToken.sol

```
/*@CTK decreaseApproval
  @tag assume_completion
  @pre _spender != 0x0
  @pre _spender != msg.sender
  @post (_subtractedValue > allowed[msg.sender][_spender]) -> (__post.allowed[msg.
      sender][_spender] == 0)
  @post (_subtractedValue <= allowed[msg.sender][_spender]) -> (__post.allowed[msg.
      sender][_spender] == allowed[msg.sender][_spender] - _subtractedValue)
*/
```

Line 343-358 in File BenepitToken.sol

```
function decreaseApproval(
  address _spender,
  uint256 _subtractedValue
)
  public
  returns (bool)
{
  uint256 oldValue = allowed[msg.sender][_spender];
  if (_subtractedValue >= oldValue) {
    allowed[msg.sender][_spender] = 0;
  } else {
    allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
  }
  emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
  return true;
}
```

✓ The code meets the specification.

Formal Verification Request 41

If method completes, integer overflow would not happen.

📅 20, Jun 2019
⏱ 12.94 ms

Line 402 in File BenepitToken.sol

```
  //@CTK NO_OVERFLOW
```

Line 410-413 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 42

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

📅 20, Jun 2019
⏱ 0.56 ms

Line 403 in File BenepitToken.sol

```
  //@CTK NO_BUF_OVERFLOW
```

Line 410-413 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 43

Method will not encounter an assertion failure.

20, Jun 2019
0.49 ms

Line 404 in File BenepitToken.sol

```
  //@CTK NO_ASF
```

Line 410-413 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 44

isOwner

20, Jun 2019
1.53 ms

Line 405-409 in File BenepitToken.sol

```
  /*@CTK isOwner
    @tag assume_completion
    @pre msg.sender == owner
    @post __post.owner == address(0)
   */
```

Line 410-413 in File BenepitToken.sol

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

✓ The code meets the specification.

Formal Verification Request 45

If method completes, integer overflow would not happen.

📅 20, Jun 2019
⏱ 40.46 ms

Line 419 in File BenepitToken.sol

```
//@CTK NO_OVERFLOW
```

Line 429-431 in File BenepitToken.sol

```
function transferOwnership(address _newOwner) public onlyOwner {
  _transferOwnership(_newOwner);
}
```

✓ The code meets the specification.

Formal Verification Request 46

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

📅 20, Jun 2019
⏱ 0.59 ms

Line 420 in File BenepitToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 429-431 in File BenepitToken.sol

```
function transferOwnership(address _newOwner) public onlyOwner {
  _transferOwnership(_newOwner);
}
```

✓ The code meets the specification.

Formal Verification Request 47

Method will not encounter an assertion failure.

📅 20, Jun 2019
⏱ 0.57 ms

Line 421 in File BenepitToken.sol

```
//@CTK NO_ASF
```

Line 429-431 in File BenepitToken.sol

```
function transferOwnership(address _newOwner) public onlyOwner {
  _transferOwnership(_newOwner);
}
```

✓ The code meets the specification.

Formal Verification Request 48

transferOwnership

📅 20, Jun 2019
⏱ 2.07 ms

Line 422-428 in File BenepitToken.sol

```
/*@CTK transferOwnership
  @tag assume_completion
  @pre msg.sender == owner
  @pre msg.sender != _newOwner
  @pre _newOwner != address(0)
  @post __post.owner == _newOwner
*/
```

Line 429-431 in File BenepitToken.sol

```
function transferOwnership(address _newOwner) public onlyOwner {
  _transferOwnership(_newOwner);
}
```

✓ The code meets the specification.

Formal Verification Request 49

If method completes, integer overflow would not happen.

📅 20, Jun 2019
⏱ 0.47 ms

Line 437 in File BenepitToken.sol

```
//@CTK NO_OVERFLOW
```

Line 447-451 in File BenepitToken.sol

```
function _transferOwnership(address _newOwner) internal {
  require(_newOwner != address(0));
  emit OwnershipTransferred(owner, _newOwner);
  owner = _newOwner;
}
```

✓ The code meets the specification.

Formal Verification Request 50

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

📅 20, Jun 2019
⏱ 0.46 ms

Line 438 in File BenepitToken.sol

```
//@CTK NO_BUF_OVERFLOW
```

Line 447-451 in File BenepitToken.sol

```
function _transferOwnership(address _newOwner) internal {
  require(_newOwner != address(0));
  emit OwnershipTransferred(owner, _newOwner);
  owner = _newOwner;
}
```

✓ The code meets the specification.

Formal Verification Request 51

Method will not encounter an assertion failure.

20, Jun 2019
0.46 ms

Line 439 in File BenepitToken.sol

```
//@CTK NO_ASF
```

Line 447-451 in File BenepitToken.sol

```
function _transferOwnership(address _newOwner) internal {
  require(_newOwner != address(0));
  emit OwnershipTransferred(owner, _newOwner);
  owner = _newOwner;
}
```

✓ The code meets the specification.

Formal Verification Request 52

_transferOwnership

20, Jun 2019
1.47 ms

Line 440-446 in File BenepitToken.sol

```
/*@CTK _transferOwnership
  @tag assume_completion
  @pre msg.sender == owner
  @pre msg.sender != _newOwner
  @pre _newOwner != address(0)
  @post __post.owner == _newOwner
*/
```

Line 447-451 in File BenepitToken.sol

```
function _transferOwnership(address _newOwner) internal {
  require(_newOwner != address(0));
  emit OwnershipTransferred(owner, _newOwner);
  owner = _newOwner;
}
```

✓ The code meets the specification.

Formal Verification Request 53

If method completes, integer overflow would not happen.

📅 20, Jun 2019
⏱ 10.6 ms

Line 460 in File BenepitToken.sol

```
  //@CTK NO_OVERFLOW
```

Line 467-470 in File BenepitToken.sol

```
  constructor() public {
    totalSupply_ = 30000000000 * 10**uint(decimals);
    balances[msg.sender] = totalSupply_;
  }
```

✓ The code meets the specification.

Formal Verification Request 54

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

📅 20, Jun 2019
⏱ 0.41 ms

Line 461 in File BenepitToken.sol

```
  //@CTK NO_BUF_OVERFLOW
```

Line 467-470 in File BenepitToken.sol

```
  constructor() public {
    totalSupply_ = 30000000000 * 10**uint(decimals);
    balances[msg.sender] = totalSupply_;
  }
```

✓ The code meets the specification.

Formal Verification Request 55

Method will not encounter an assertion failure.

📅 20, Jun 2019
⏱ 0.43 ms

Line 462 in File BenepitToken.sol

```
  //@CTK NO_ASF
```

Line 467-470 in File BenepitToken.sol

```
  constructor() public {
    totalSupply_ = 30000000000 * 10**uint(decimals);
    balances[msg.sender] = totalSupply_;
  }
```

✓ The code meets the specification.

Formal Verification Request 56

BenepitToken

📅 20, Jun 2019
⏱ 1.89 ms

Line 463-466 in File BenepitToken.sol

```
/*@CTK BenepitToken
  @tag assume_completion
  @post __post.balances[msg.sender] == __post.totalSupply_
*/
```

Line 467-470 in File BenepitToken.sol

```
constructor() public {
  totalSupply_ = 30000000000 * 10**uint(decimals);
  balances[msg.sender] = totalSupply_;
}
```

✓ The code meets the specification.

Source Code with CertiK Labels

File BenepitToken.sol

```
pragma solidity 0.4.24;


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


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

  /**
  * @dev Multiplies two numbers, throws on overflow.
  */
  //@CTK FAIL NO_ASF
  /*@CTK "SafeMath mul"
    @post ((_a > 0) && (((_a * _b) / _a) != _b)) == (__reverted)
    @post !__reverted -> c == _a * _b
    @post !__reverted == !__has_overflow
    @post !__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.
    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 !__reverted -> !(__has_assertion_failure)
    @post !(__has_buf_overflow)
  */
  function add(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
    c = _a + _b;
    assert(c >= _a);
    return c;
  }
}



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

  mapping(address => uint256) internal balances;

  uint256 internal 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(_value <= balances[msg.sender]);
  require(_to != address(0));

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

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

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



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

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


  /**
   * @dev Transfer tokens from one address to another
   * @param _from address The address which you want to send tokens from
   * @param _to address The address which you want to transfer to
   * @param _value uint256 the amount of tokens to be transferred
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK FAIL NO_ASF
  /*@CTK "transferFrom"
    @tag assume_completion
    @pre (_to) != (address(0))
    @pre (_value) <= (balances[_from])
    @pre (_value) <= (allowed[_from][msg.sender])
    @post (_from != _to) -> (__post.balances[_to] == (balances[_to] + _value))
    @post (_from != _to) -> (__post.balances[_from] == (balances[_from] - _value))
    @post (_from == _to) -> (__post.balances[_to] == balances[_to])
    @post (_from == _to) -> (__post.balances[_from] == balances[_from])
    @post (__post.allowed[_from][msg.sender]) == (allowed[_from][msg.sender] - _value)
    @post (__return) == (true)
  */
  function transferFrom(
    address _from,
    address _to,
    uint256 _value
  )
    public
    returns (bool)
  {
    require(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);
    require(_to != address(0));

    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
    @tag assume_completion
    @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
    @tag assume_completion
    @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
    @tag assume_completion
    @pre _spender != 0x0
    @pre _spender != msg.sender
    @post (__post.allowed[msg.sender][_spender]) == (allowed[msg.sender][_spender] +
        _addedValue)
    @post (__return) == (true)
  */
  function increaseApproval(
    address _spender,
    uint256 _addedValue
  )
    public
    returns (bool)
  {
    allowed[msg.sender][_spender] = (
      allowed[msg.sender][_spender].add(_addedValue));
    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    return true;
  }

  /**
   * @dev Decrease the amount of tokens that an owner allowed to a spender.
   * approve should be called when allowed[_spender] == 0. To decrement
   * allowed value is better to use this function to avoid 2 calls (and wait until
   * the first transaction is mined)
   * From MonolithDAO Token.sol
   * @param _spender The address which will spend the funds.
   * @param _subtractedValue The amount of tokens to decrease the allowance by.
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK decreaseApproval
    @tag assume_completion
    @pre _spender != 0x0
    @pre _spender != msg.sender
    @post (_subtractedValue > allowed[msg.sender][_spender]) -> (__post.allowed[msg.
        sender][_spender] == 0)
    @post (_subtractedValue <= allowed[msg.sender][_spender]) -> (__post.allowed[msg.
        sender][_spender] == allowed[msg.sender][_spender] - _subtractedValue)
  */
  function decreaseApproval(
    address _spender,
    uint256 _subtractedValue
  )
    public
    returns (bool)
  {
    uint256 oldValue = allowed[msg.sender][_spender];
    if (_subtractedValue >= oldValue) {
      allowed[msg.sender][_spender] = 0;
    } else {
      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
    }
    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    return true;
  }

}



/**
 * @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.
   * @notice Renouncing to ownership will leave the contract without an owner.
   * It will not be possible to call the functions with the `onlyOwner`
   * modifier anymore.
   */
    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK isOwner
      @tag assume_completion
      @pre 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.
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK transferOwnership
    @tag assume_completion
    @pre msg.sender == owner
    @pre msg.sender != _newOwner
    @pre _newOwner != address(0)
    @post __post.owner == _newOwner
  */
  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 NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK _transferOwnership
    @tag assume_completion
    @pre msg.sender == owner
    @pre msg.sender != _newOwner
    @pre _newOwner != address(0)
    @post __post.owner == _newOwner
  */
  function _transferOwnership(address _newOwner) internal {
    require(_newOwner != address(0));
    emit OwnershipTransferred(owner, _newOwner);
    owner = _newOwner;
  }
}


contract BenepitToken is StandardToken, Ownable {
    string public name = "Benepit";
    string public symbol = "BNP";
    uint8 public decimals = 18;

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK BenepitToken
      @tag assume_completion
      @post __post.balances[msg.sender] == __post.totalSupply_
    */
    constructor() public {
      totalSupply_ = 30000000000 * 10**uint(decimals);
      balances[msg.sender] = totalSupply_;
    }
}
```