CertiK Verification Report For MultiVAC

Request Date: 2019-03-28
Revision Date: 2019-03-28

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 MultiVAC(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 CertiKs prior written consent.

Summary

This audit report summarises the smart contract verification service requested by MultiVAC. The goal of this security audit is to guarantee that the audited smart contracts are robust enough to avoid any potential security 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 time of the audit.

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Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	1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

Insecure Pragma Version:

- The source code is using `pragma solidity ^0.4.22;`, only compiler version 0.4.25 is absolutely safe to use.

Redundant ValidAddress check:

- modifier `validAddress` is not necessarily needed, or could `assert` to replace `require` to indicate it will never happen.

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 MultiVAC.sol

```
pragma solidity ^0.4.22;

contract MultiVACToken {
    string public name = "MultiVAC";  //  token name
    string public symbol = "MTV";       //  token symbol
    uint256 public decimals = 18;        //  token digit
    mapping (address => uint256) public balanceOf;
    mapping (address => mapping (address => uint256)) public allowance;
    uint256 public totalSupply = 0;
    bool public stopped = false;
    uint256 constant initSupply = 10**10;
    address owner = address(0);

    event Transfer(address indexed _from, address indexed _to, uint256 _value);
    event Approval(address indexed _owner, address indexed _spender, uint256 _value);

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

    modifier isRunning {
        require(!stopped);
        _;
    }

    modifier validAddress {
        require(address(0) != msg.sender);
        _;
    }

    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "constructor"
      @post __post.balanceOf[msg.sender] == __post.totalSupply
      @post __post.owner == msg.sender
     */
    constructor() public {
        owner = msg.sender;
        totalSupply = initSupply * (10 ** decimals);
        balanceOf[msg.sender] = totalSupply;
        emit Transfer(address(0), msg.sender, totalSupply);
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK transfer
      @tag assume_completion
      @pre msg.sender != _to
      @post _to != 0
      @post balanceOf[msg.sender] >= _value
      @post balanceOf[_to] + _value >= balanceOf[_to]
      @post __post.balanceOf[msg.sender] == balanceOf[msg.sender] - _value
```

```
     @post __post.balanceOf[_to] == balanceOf[_to] + _value
    */
   /*@CTK transferToSame
     @tag assume_completion
     @pre msg.sender == _to
     @post __post.balanceOf[msg.sender] == balanceOf[msg.sender]
    */
   function transfer(address _to, uint256 _value) public isRunning validAddress
       returns (bool success) {
       require(_to != address(0));
       require(balanceOf[msg.sender] >= _value);
       require(balanceOf[_to] + _value >= balanceOf[_to]);
       balanceOf[msg.sender] -= _value;
       balanceOf[_to] += _value;
       emit Transfer(msg.sender, _to, _value);
       return true;
   }

   //@CTK NO_OVERFLOW
   //@CTK NO_BUF_OVERFLOW
   //@CTK NO_ASF
   function transferFrom(address _from, address _to, uint256 _value) public isRunning
        validAddress returns (bool success) {
       require(_to != address(0));
       require(balanceOf[_from] >= _value);
       require(balanceOf[_to] + _value >= balanceOf[_to]);
       require(allowance[_from][msg.sender] >= _value);
       balanceOf[_to] += _value;
       balanceOf[_from] -= _value;
       allowance[_from][msg.sender] -= _value;
       emit Transfer(_from, _to, _value);
       return true;
   }

   //@CTK NO_OVERFLOW
   //@CTK NO_BUF_OVERFLOW
   //@CTK NO_ASF
   /*@CTK approve1
     @pre stopped == false
     @pre msg.sender != 0x0
     @pre allowance[msg.sender][_spender] == 0
     @pre _value > 0
     @post __post.allowance[msg.sender][_spender] == _value
    */
   /*@CTK approve2
     @pre stopped == false
     @pre msg.sender != 0x0
     @pre allowance[msg.sender][_spender] > 0
     @pre _value > 0
     @post __reverted
    */
   /*@CTK approve3
     @pre stopped == false
     @pre msg.sender != 0x0
     @pre allowance[msg.sender][_spender] == 0
     @pre _value == 0
     @post __post.allowance[msg.sender][_spender] == _value
    */
```

```
    /*@CTK approve4
      @pre stopped == false
      @pre msg.sender != 0x0
      @pre allowance[msg.sender][_spender] > 0
      @pre _value == 0
      @post __post.allowance[msg.sender][_spender] == _value
     */
    function approve(address _spender, uint256 _value) public isRunning validAddress
        returns (bool success) {
        require(_value == 0 || allowance[msg.sender][_spender] == 0);
        allowance[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value);
        return true;
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK stop
      @post owner != msg.sender -> __reverted
      @post owner == msg.sender -> !__reverted && __post.stopped == true
     */
    function stop() public isOwner {
        stopped = true;
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK start
      @post owner != msg.sender -> __reverted
      @post owner == msg.sender -> !__reverted && __post.stopped == false
     */
    function start() public isOwner {
        stopped = false;
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK setName
      @post owner != msg.sender -> __reverted
      @post owner == msg.sender -> !__reverted && __post.name == _name
     */
    function setName(string _name) public isOwner {
        name = _name;
    }

    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK burn
      @pre msg.sender != 0x0
      @pre stopped == false
      @pre balanceOf[address(0)] + _value >= balanceOf[address(0)]
      @pre balanceOf[msg.sender] >= _value
      @post __post.balanceOf[msg.sender] == balanceOf[msg.sender] - _value
      @post __post.balanceOf[address(0)] == balanceOf[address(0)] + _value
      @post !__has_overflow
```

```
     */
    function burn(uint256 _value) public isRunning {
        require(balanceOf[msg.sender] >= _value);
        balanceOf[msg.sender] -= _value;
        balanceOf[address(0)] += _value;
        emit Transfer(msg.sender, address(0), _value);
    }

    function () public payable {
        revert();
    }
}
```

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 MultiVAC.sol

```
1 pragma solidity ^0.4.22;
```

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

Formal Verification Request 1

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

28, Mar 2019
21.5 ms

Line 32 in File MultiVAC.sol

```
32    //@CTK NO_BUF_OVERFLOW
```

Line 38-43 in File MultiVAC.sol

```
38    constructor() public {
39        owner = msg.sender;
40        totalSupply = initSupply * (10 ** decimals);
41        balanceOf[msg.sender] = totalSupply;
42        emit Transfer(address(0), msg.sender, totalSupply);
43    }
```

The code meets the specification

Formal Verification Request 2

Method will not encounter an assertion failure.

28, Mar 2019
0.53 ms

Line 33 in File MultiVAC.sol

```
33    //@CTK NO_ASF
```

Line 38-43 in File MultiVAC.sol

```
38    constructor() public {
39        owner = msg.sender;
40        totalSupply = initSupply * (10 ** decimals);
41        balanceOf[msg.sender] = totalSupply;
42        emit Transfer(address(0), msg.sender, totalSupply);
43    }
```

The code meets the specification

Formal Verification Request 3

constructor

28, Mar 2019
6.34 ms

Line 34-37 in File MultiVAC.sol

```
34    /*@CTK "constructor"
35      @post __post.balanceOf[msg.sender] == __post.totalSupply
36      @post __post.owner == msg.sender
37     */
```

Line 38-43 in File MultiVAC.sol

```
38    constructor() public {
39        owner = msg.sender;
40        totalSupply = initSupply * (10 ** decimals);
41        balanceOf[msg.sender] = totalSupply;
42        emit Transfer(address(0), msg.sender, totalSupply);
43    }
```

✅ The code meets the specification

Formal Verification Request 4

If method completes, integer overflow would not happen.

📅 28, Mar 2019
⏱ 156.23 ms

Line 45 in File MultiVAC.sol

```
45    //@CTK NO_OVERFLOW
```

Line 62-70 in File MultiVAC.sol

```
62    function transfer(address _to, uint256 _value) public isRunning validAddress
          returns (bool success) {
63        require(_to != address(0));
64        require(balanceOf[msg.sender] >= _value);
65        require(balanceOf[_to] + _value >= balanceOf[_to]);
66        balanceOf[msg.sender] -= _value;
67        balanceOf[_to] += _value;
68        emit Transfer(msg.sender, _to, _value);
69        return true;
70    }
```

✅ The code meets the specification

Formal Verification Request 5

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

📅 28, Mar 2019
⏱ 13.39 ms

Line 46 in File MultiVAC.sol

```
46    //@CTK NO_BUF_OVERFLOW
```

Line 62-70 in File MultiVAC.sol

```
62    function transfer(address _to, uint256 _value) public isRunning validAddress
          returns (bool success) {
63        require(_to != address(0));
64        require(balanceOf[msg.sender] >= _value);
65        require(balanceOf[_to] + _value >= balanceOf[_to]);
66        balanceOf[msg.sender] -= _value;
67        balanceOf[_to] += _value;
68        emit Transfer(msg.sender, _to, _value);
69        return true;
70    }
```

The code meets the specification

Formal Verification Request 6

Method will not encounter an assertion failure.

28, Mar 2019
16.86 ms

Line 47 in File MultiVAC.sol

```
47    //@CTK NO_ASF
```

Line 62-70 in File MultiVAC.sol

```
62    function transfer(address _to, uint256 _value) public isRunning validAddress
          returns (bool success) {
63        require(_to != address(0));
64        require(balanceOf[msg.sender] >= _value);
65        require(balanceOf[_to] + _value >= balanceOf[_to]);
66        balanceOf[msg.sender] -= _value;
67        balanceOf[_to] += _value;
68        emit Transfer(msg.sender, _to, _value);
69        return true;
70    }
```

The code meets the specification

Formal Verification Request 7

transfer

28, Mar 2019
262.22 ms

Line 48-56 in File MultiVAC.sol

```
48    /*@CTK transfer
49      @tag assume_completion
50      @pre msg.sender != _to
51      @post _to != 0
52      @post balanceOf[msg.sender] >= _value
53      @post balanceOf[_to] + _value >= balanceOf[_to]
54      @post __post.balanceOf[msg.sender] == balanceOf[msg.sender] - _value
```

```
55      @post __post.balanceOf[_to] == balanceOf[_to] + _value
56     */
```

Line 62-70 in File MultiVAC.sol

```
62    function transfer(address _to, uint256 _value) public isRunning validAddress
          returns (bool success) {
63        require(_to != address(0));
64        require(balanceOf[msg.sender] >= _value);
65        require(balanceOf[_to] + _value >= balanceOf[_to]);
66        balanceOf[msg.sender] -= _value;
67        balanceOf[_to] += _value;
68        emit Transfer(msg.sender, _to, _value);
69        return true;
70    }
```

✅ The code meets the specification

Formal Verification Request 8

transferToSame

📅 28, Mar 2019
⏱ 54.64 ms

Line 57-61 in File MultiVAC.sol

```
57    /*@CTK transferToSame
58      @tag assume_completion
59      @pre msg.sender == _to
60      @post __post.balanceOf[msg.sender] == balanceOf[msg.sender]
61     */
```

Line 62-70 in File MultiVAC.sol

```
62    function transfer(address _to, uint256 _value) public isRunning validAddress
          returns (bool success) {
63        require(_to != address(0));
64        require(balanceOf[msg.sender] >= _value);
65        require(balanceOf[_to] + _value >= balanceOf[_to]);
66        balanceOf[msg.sender] -= _value;
67        balanceOf[_to] += _value;
68        emit Transfer(msg.sender, _to, _value);
69        return true;
70    }
```

✅ The code meets the specification

Formal Verification Request 9

If method completes, integer overflow would not happen.

📅 28, Mar 2019
⏱ 197.28 ms

Line 72 in File MultiVAC.sol

```
72    //@CTK NO_OVERFLOW
```

Line 75-85 in File MultiVAC.sol

```
75    function transferFrom(address _from, address _to, uint256 _value) public isRunning
          validAddress returns (bool success) {
76        require(_to != address(0));
77        require(balanceOf[_from] >= _value);
78        require(balanceOf[_to] + _value >= balanceOf[_to]);
79        require(allowance[_from][msg.sender] >= _value);
80        balanceOf[_to] += _value;
81        balanceOf[_from] -= _value;
82        allowance[_from][msg.sender] -= _value;
83        emit Transfer(_from, _to, _value);
84        return true;
85    }
```

✅ The code meets the specification

Formal Verification Request 10

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

📅 28, Mar 2019
⏱ 25.6 ms

Line 73 in File MultiVAC.sol

```
73    //@CTK NO_BUF_OVERFLOW
```

Line 75-85 in File MultiVAC.sol

```
75    function transferFrom(address _from, address _to, uint256 _value) public isRunning
          validAddress returns (bool success) {
76        require(_to != address(0));
77        require(balanceOf[_from] >= _value);
78        require(balanceOf[_to] + _value >= balanceOf[_to]);
79        require(allowance[_from][msg.sender] >= _value);
80        balanceOf[_to] += _value;
81        balanceOf[_from] -= _value;
82        allowance[_from][msg.sender] -= _value;
83        emit Transfer(_from, _to, _value);
84        return true;
85    }
```

✅ The code meets the specification

Formal Verification Request 11

Method will not encounter an assertion failure.

📅 28, Mar 2019
⏱ 37.32 ms

Line 74 in File MultiVAC.sol

```
74    //@CTK NO_ASF
```

Line 75-85 in File MultiVAC.sol

```
75    function transferFrom(address _from, address _to, uint256 _value) public isRunning
          validAddress returns (bool success) {
76        require(_to != address(0));
77        require(balanceOf[_from] >= _value);
78        require(balanceOf[_to] + _value >= balanceOf[_to]);
79        require(allowance[_from][msg.sender] >= _value);
80        balanceOf[_to] += _value;
81        balanceOf[_from] -= _value;
82        allowance[_from][msg.sender] -= _value;
83        emit Transfer(_from, _to, _value);
84        return true;
85    }
```

The code meets the specification

Formal Verification Request 12

If method completes, integer overflow would not happen.

28, Mar 2019
46.54 ms

Line 87 in File MultiVAC.sol

```
87    //@CTK NO_OVERFLOW
```

Line 118-123 in File MultiVAC.sol

```
118   function approve(address _spender, uint256 _value) public isRunning validAddress
          returns (bool success) {
119       require(_value == 0 || allowance[msg.sender][_spender] == 0);
120       allowance[msg.sender][_spender] = _value;
121       emit Approval(msg.sender, _spender, _value);
122       return true;
123   }
```

The code meets the specification

Formal Verification Request 13

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

28, Mar 2019
3.01 ms

Line 88 in File MultiVAC.sol

```
88    //@CTK NO_BUF_OVERFLOW
```

Line 118-123 in File MultiVAC.sol

```
118    function approve(address _spender, uint256 _value) public isRunning validAddress
           returns (bool success) {
119        require(_value == 0 || allowance[msg.sender][_spender] == 0);
120        allowance[msg.sender][_spender] = _value;
121        emit Approval(msg.sender, _spender, _value);
122        return true;
123    }
```

The code meets the specification

Formal Verification Request 14

Method will not encounter an assertion failure.

28, Mar 2019
3.38 ms

Line 89 in File MultiVAC.sol

```
89    //@CTK NO_ASF
```

Line 118-123 in File MultiVAC.sol

```
118    function approve(address _spender, uint256 _value) public isRunning validAddress
           returns (bool success) {
119        require(_value == 0 || allowance[msg.sender][_spender] == 0);
120        allowance[msg.sender][_spender] = _value;
121        emit Approval(msg.sender, _spender, _value);
122        return true;
123    }
```

The code meets the specification

Formal Verification Request 15

approve1

28, Mar 2019
3.98 ms

Line 90-96 in File MultiVAC.sol

```
90    /*@CTK approve1
91      @pre stopped == false
92      @pre msg.sender != 0x0
93      @pre allowance[msg.sender][_spender] == 0
94      @pre _value > 0
95      @post __post.allowance[msg.sender][_spender] == _value
96     */
```

Line 118-123 in File MultiVAC.sol

```
118    function approve(address _spender, uint256 _value) public isRunning validAddress
           returns (bool success) {
119        require(_value == 0 || allowance[msg.sender][_spender] == 0);
```

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

The code meets the specification

Formal Verification Request 16

approve2

28, Mar 2019
3.36 ms

Line 97-103 in File MultiVAC.sol

```
97    /*@CTK approve2
98      @pre stopped == false
99      @pre msg.sender != 0x0
100     @pre allowance[msg.sender][_spender] > 0
101     @pre _value > 0
102     @post __reverted
103    */
```

Line 118-123 in File MultiVAC.sol

```
118   function approve(address _spender, uint256 _value) public isRunning validAddress
          returns (bool success) {
119       require(_value == 0 || allowance[msg.sender][_spender] == 0);
120       allowance[msg.sender][_spender] = _value;
121       emit Approval(msg.sender, _spender, _value);
122       return true;
123   }
```

The code meets the specification

Formal Verification Request 17

approve3

28, Mar 2019
3.3 ms

Line 104-110 in File MultiVAC.sol

```
104   /*@CTK approve3
105     @pre stopped == false
106     @pre msg.sender != 0x0
107     @pre allowance[msg.sender][_spender] == 0
108     @pre _value == 0
109     @post __post.allowance[msg.sender][_spender] == _value
110    */
```

Line 118-123 in File MultiVAC.sol

```
118    function approve(address _spender, uint256 _value) public isRunning validAddress
           returns (bool success) {
119        require(_value == 0 || allowance[msg.sender][_spender] == 0);
120        allowance[msg.sender][_spender] = _value;
121        emit Approval(msg.sender, _spender, _value);
122        return true;
123    }
```

The code meets the specification

Formal Verification Request 18

approve4

28, Mar 2019
4.07 ms

Line 111-117 in File MultiVAC.sol

```
111    /*@CTK approve4
112      @pre stopped == false
113      @pre msg.sender != 0x0
114      @pre allowance[msg.sender][_spender] > 0
115      @pre _value == 0
116      @post __post.allowance[msg.sender][_spender] == _value
117     */
```

Line 118-123 in File MultiVAC.sol

```
118    function approve(address _spender, uint256 _value) public isRunning validAddress
           returns (bool success) {
119        require(_value == 0 || allowance[msg.sender][_spender] == 0);
120        allowance[msg.sender][_spender] = _value;
121        emit Approval(msg.sender, _spender, _value);
122        return true;
123    }
```

The code meets the specification

Formal Verification Request 19

If method completes, integer overflow would not happen.

28, Mar 2019
21.42 ms

Line 125 in File MultiVAC.sol

```
125    //@CTK NO_OVERFLOW
```

Line 132-134 in File MultiVAC.sol

```
132    function stop() public isOwner {
133        stopped = true;
134    }
```

The code meets the specification

Formal Verification Request 20

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

28, Mar 2019
0.64 ms

Line 126 in File MultiVAC.sol

```
126    //@CTK NO_BUF_OVERFLOW
```

Line 132-134 in File MultiVAC.sol

```
132    function stop() public isOwner {
133        stopped = true;
134    }
```

The code meets the specification

Formal Verification Request 21

Method will not encounter an assertion failure.

28, Mar 2019
1.18 ms

Line 127 in File MultiVAC.sol

```
127    //@CTK NO_ASF
```

Line 132-134 in File MultiVAC.sol

```
132    function stop() public isOwner {
133        stopped = true;
134    }
```

The code meets the specification

Formal Verification Request 22

stop

28, Mar 2019
3.05 ms

Line 128-131 in File MultiVAC.sol

```
128    /*@CTK stop
129      @post owner != msg.sender -> __reverted
130      @post owner == msg.sender -> !__reverted && __post.stopped == true
131     */
```

Line 132-134 in File MultiVAC.sol

```
132    function stop() public isOwner {
133        stopped = true;
134    }
```

The code meets the specification

Formal Verification Request 23

If method completes, integer overflow would not happen.

28, Mar 2019
19.47 ms

Line 136 in File MultiVAC.sol

```
136    //@CTK NO_OVERFLOW
```

Line 143-145 in File MultiVAC.sol

```
143    function start() public isOwner {
144        stopped = false;
145    }
```

The code meets the specification

Formal Verification Request 24

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

28, Mar 2019
0.5 ms

Line 137 in File MultiVAC.sol

```
137    //@CTK NO_BUF_OVERFLOW
```

Line 143-145 in File MultiVAC.sol

```
143    function start() public isOwner {
144        stopped = false;
145    }
```

The code meets the specification

Formal Verification Request 25

Method will not encounter an assertion failure.

28, Mar 2019
0.51 ms

Line 138 in File MultiVAC.sol

```
138    //@CTK NO_ASF
```

Line 143-145 in File MultiVAC.sol

```
143     function start() public isOwner {
144         stopped = false;
145     }
```

The code meets the specification

Formal Verification Request 26

start

28, Mar 2019
2.8 ms

Line 139-142 in File MultiVAC.sol

```
139     /*@CTK start
140       @post owner != msg.sender -> __reverted
141       @post owner == msg.sender -> !__reverted && __post.stopped == false
142      */
```

Line 143-145 in File MultiVAC.sol

```
143     function start() public isOwner {
144         stopped = false;
145     }
```

The code meets the specification

Formal Verification Request 27

If method completes, integer overflow would not happen.

28, Mar 2019
21.99 ms

Line 147 in File MultiVAC.sol

```
147     //@CTK NO_OVERFLOW
```

Line 154-156 in File MultiVAC.sol

```
154     function setName(string _name) public isOwner {
155         name = _name;
156     }
```

The code meets the specification

Formal Verification Request 28

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

28, Mar 2019
0.51 ms

Line 148 in File MultiVAC.sol

```
148    //@CTK NO_BUF_OVERFLOW
```

Line 154-156 in File MultiVAC.sol

```
154    function setName(string _name) public isOwner {
155        name = _name;
156    }
```

The code meets the specification

Formal Verification Request 29

Method will not encounter an assertion failure.

28, Mar 2019
0.53 ms

Line 149 in File MultiVAC.sol

```
149    //@CTK NO_ASF
```

Line 154-156 in File MultiVAC.sol

```
154    function setName(string _name) public isOwner {
155        name = _name;
156    }
```

The code meets the specification

Formal Verification Request 30

setName

28, Mar 2019
2.07 ms

Line 150-153 in File MultiVAC.sol

```
150    /*@CTK setName
151      @post owner != msg.sender -> __reverted
152      @post owner == msg.sender -> !__reverted && __post.name == _name
153     */
```

Line 154-156 in File MultiVAC.sol

```
154    function setName(string _name) public isOwner {
155        name = _name;
156    }
```

The code meets the specification

Formal Verification Request 31

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

28, Mar 2019
39.42 ms

Line 158 in File MultiVAC.sol

```
158    //@CTK NO_BUF_OVERFLOW
```

Line 169-174 in File MultiVAC.sol

```
169    function burn(uint256 _value) public isRunning {
170        require(balanceOf[msg.sender] >= _value);
171        balanceOf[msg.sender] -= _value;
172        balanceOf[address(0)] += _value;
173        emit Transfer(msg.sender, address(0), _value);
174    }
```

The code meets the specification

Formal Verification Request 32

Method will not encounter an assertion failure.

28, Mar 2019
1.26 ms

Line 159 in File MultiVAC.sol

```
159    //@CTK NO_ASF
```

Line 169-174 in File MultiVAC.sol

```
169    function burn(uint256 _value) public isRunning {
170        require(balanceOf[msg.sender] >= _value);
171        balanceOf[msg.sender] -= _value;
172        balanceOf[address(0)] += _value;
173        emit Transfer(msg.sender, address(0), _value);
174    }
```

The code meets the specification

Formal Verification Request 33

burn

28, Mar 2019
112.26 ms

Line 160-168 in File MultiVAC.sol

```
160   /*@CTK burn
161     @pre msg.sender != 0x0
162     @pre stopped == false
163     @pre balanceOf[address(0)] + _value >= balanceOf[address(0)]
164     @pre balanceOf[msg.sender] >= _value
165     @post __post.balanceOf[msg.sender] == balanceOf[msg.sender] - _value
166     @post __post.balanceOf[address(0)] == balanceOf[address(0)] + _value
167     @post !__has_overflow
168    */
```

Line 169-174 in File MultiVAC.sol

```
169   function burn(uint256 _value) public isRunning {
170       require(balanceOf[msg.sender] >= _value);
171       balanceOf[msg.sender] -= _value;
172       balanceOf[address(0)] += _value;
173       emit Transfer(msg.sender, address(0), _value);
174   }
```

✅ The code meets the specification