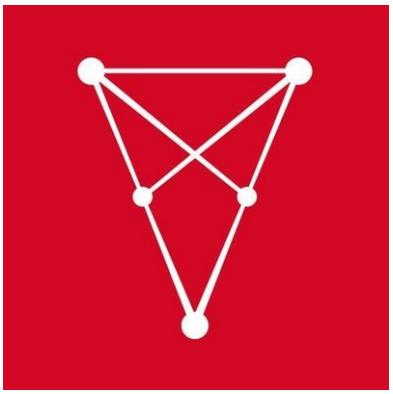
# CertiK Verification Report
# For chiliZ



Request Date: 2018-11-16
Revision Date: 2018-11-22

# 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 chiliZ(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.

PASS

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

*Nov 22, 2018*


## Summary

This is the report for smart contract verification service requestd by chiliZ. 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. | 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 Compiler Version:

- The source code is using `pragma solidity ^0.4.24`, while `0.4.24` is susceptible to bug *ExpExponentCleanup* and *EventStructWrongData*.
- It is recommended to use latest and fixed compiler version.

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 chiliZ.sol

```
pragma solidity ^0.4.24;

// File: contracts/Chiliz-with-imports.sol

import "./openzeppelin-solidity/contracts/token/ERC20/ERC20.sol";
import "./openzeppelin-solidity/contracts/token/ERC20/ERC20Detailed.sol";
import "./openzeppelin-solidity/contracts/token/ERC20/ERC20Pausable.sol";
import "./openzeppelin-solidity/contracts/token/ERC20/SafeERC20.sol";

contract chiliZ is ERC20, ERC20Detailed, ERC20Pausable {
  using SafeERC20 for ERC20;

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  constructor()
    ERC20()
    ERC20Detailed("chiliZ", "CHZ", 18)
    ERC20Pausable()
  public
  {
    _mint(msg.sender, 8888888888 * (uint256(10) ** 18));
  }
}
```

File openzeppelin-solidity/contracts/access/Roles.sol

```
pragma solidity ^0.4.24;

/**
 * @title Roles
 * @dev Library for managing addresses assigned to a Role.
 */
library Roles {
  struct Role {
    mapping (address => bool) bearer;
  }

  /**
   * @dev give an account access to this role
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  /*@CTK "Roles add correctness"
    @post account == 0x0 -> __reverted
    @post role.bearer[account] -> __reverted
    @post account != 0x0 && !role.bearer[account] -> !__reverted
   */
  function add(Role storage role, address account) internal {
    require(account != address(0));
    require(!role.bearer[account]);

    role.bearer[account] = true;
  }
```

```
29
30    /**
31     * @dev remove an account's access to this role
32     */
33    //@CTK NO_OVERFLOW
34    //@CTK NO_BUF_OVERFLOW
35    //@CTK NO_ASF
36    /*@CTK "Roles add correctness"
37      @post account == 0x0 -> __reverted
38      @post !role.bearer[account] -> __reverted
39      @post account != 0x0 && role.bearer[account] -> !__reverted
40     */
41    function remove(Role storage role, address account) internal {
42      require(account != address(0));
43      require(role.bearer[account]);
44
45      role.bearer[account] = false;
46    }
47
48    /**
49     * @dev check if an account has this role
50     * @return bool
51     */
52    //@CTK NO_OVERFLOW
53    //@CTK NO_BUF_OVERFLOW
54    //@CTK NO_ASF
55    /*@CTK "Roles has correctness"
56      @post account == 0x0 -> __reverted
57      @post account != 0x0 -> (!__reverted) && (__return == role.bearer[account])
58     */
59    function has(Role storage role, address account)
60      internal
61      view
62      returns (bool)
63    {
64      require(account != address(0));
65      return role.bearer[account];
66    }
67  }
```

File openzeppelin-solidity/contracts/access/roles/PauserRole.sol

```
1   pragma solidity ^0.4.24;
2
3   import "../Roles.sol";
4
5   contract PauserRole {
6     using Roles for Roles.Role;
7
8     event PauserAdded(address indexed account);
9     event PauserRemoved(address indexed account);
10
11    Roles.Role private pausers;
12
13    //@CTK NO_OVERFLOW
14    //@CTK NO_BUF_OVERFLOW
15    //@CTK NO_ASF
16    /*@CTK "PauserRole constructor correctness"
17      @post msg.sender == 0x0 -> __reverted
```

```
18    @post pausers.bearer[msg.sender] -> __reverted
19    @post msg.sender != 0x0 && !pausers.bearer[msg.sender]
20          -> !__reverted && __post.pausers.bearer[msg.sender]
21   */
22  constructor() internal {
23    _addPauser(msg.sender);
24  }
25
26  modifier onlyPauser() {
27    require(isPauser(msg.sender));
28    _;
29  }
30
31  //@CTK NO_OVERFLOW
32  //@CTK NO_BUF_OVERFLOW
33  //@CTK NO_ASF
34  /*@CTK "isBurner correctness"
35    @post account == 0x0 -> __reverted
36    @post account != 0x0 -> !__reverted && __return == pausers.bearer[account]
37   */
38  function isPauser(address account) public view returns (bool) {
39    return pausers.has(account);
40  }
41
42  //@CTK NO_OVERFLOW
43  //@CTK NO_BUF_OVERFLOW
44  //@CTK NO_ASF
45  /*@CTK "_addPauser correctness"
46    @post account == 0x0 -> __reverted
47    @post msg.sender == 0x0 -> __reverted
48    @post pausers.bearer[account] -> __reverted
49    @post !pausers.bearer[msg.sender] -> __reverted
50    @post account != 0x0 && !pausers.bearer[account]
51          && msg.sender != 0x0 && pausers.bearer[msg.sender]
52          -> !__reverted && __post.pausers.bearer[account]
53   */
54  function addPauser(address account) public onlyPauser {
55    _addPauser(account);
56  }
57
58  //@CTK NO_OVERFLOW
59  //@CTK NO_BUF_OVERFLOW
60  //@CTK NO_ASF
61  /*@CTK "renouncePauser correctness"
62    @post msg.sender == 0x0 -> __reverted
63    @post !pausers.bearer[msg.sender] -> __reverted
64    @post msg.sender != 0x0 && pausers.bearer[msg.sender]
65          -> !__reverted && !__post.pausers.bearer[msg.sender]
66   */
67  function renouncePauser() public {
68    _removePauser(msg.sender);
69  }
70
71  //@CTK NO_OVERFLOW
72  //@CTK NO_BUF_OVERFLOW
73  //@CTK NO_ASF
74  /*@CTK "_addPauser correctness"
75    @post account == 0x0 -> __reverted
```

```
76      @post pausers.bearer[account] -> __reverted
77      @post account != 0x0 && !pausers.bearer[account]
78            -> !__reverted && __post.pausers.bearer[account]
79     */
80    function _addPauser(address account) internal {
81      pausers.add(account);
82      emit PauserAdded(account);
83    }
84
85    //@CTK NO_OVERFLOW
86    //@CTK NO_BUF_OVERFLOW
87    //@CTK NO_ASF
88    /*@CTK "_removePauser correctness"
89      @post account == 0x0 -> __reverted
90      @post !pausers.bearer[account] -> __reverted
91      @post account != 0x0 && pausers.bearer[account]
92            -> !__reverted && !__post.pausers.bearer[account]
93     */
94    function _removePauser(address account) internal {
95      pausers.remove(account);
96      emit PauserRemoved(account);
97    }
98  }
```

File openzeppelin-solidity/contracts/lifecycle/Pausable.sol

```
1   pragma solidity ^0.4.24;
2
3   import "../access/roles/PauserRole.sol";
4
5   /**
6    * @title Pausable
7    * @dev Base contract which allows children to implement an emergency stop mechanism.
8    */
9   contract Pausable is PauserRole {
10    event Paused(address account);
11    event Unpaused(address account);
12
13    bool private _paused;
14
15    //@CTK NO_OVERFLOW
16    //@CTK NO_BUF_OVERFLOW
17    //@CTK NO_ASF
18    /*@CTK "Pausable constructor correctness"
19      @post __post._paused == false
20     */
21    constructor() internal {
22      _paused = false;
23    }
24
25    /**
26     * @return true if the contract is paused, false otherwise.
27     */
28    //@CTK NO_OVERFLOW
29    //@CTK NO_BUF_OVERFLOW
30    //@CTK NO_ASF
31    /*@CTK "Pausable paused correctness"
32      @post __return == _paused
33     */
```

```
34    function paused() public view returns(bool) {
35      return _paused;
36    }
37
38    /**
39     * @dev Modifier to make a function callable only when the contract is not paused.
40     */
41    modifier whenNotPaused() {
42      require(!_paused);
43      _;
44    }
45
46    /**
47     * @dev Modifier to make a function callable only when the contract is paused.
48     */
49    modifier whenPaused() {
50      require(_paused);
51      _;
52    }
53
54    /**
55     * @dev called by the owner to pause, triggers stopped state
56     */
57    //@CTK NO_OVERFLOW
58    //@CTK NO_BUF_OVERFLOW
59    //@CTK NO_ASF
60    /*@CTK "Pausable pause correctness"
61      @post _paused -> __reverted
62      @post msg.sender == 0x0 -> __reverted
63      @post !pausers.bearer[msg.sender] -> __reverted
64      @post !_paused && msg.sender != 0x0 && pausers.bearer[msg.sender]
65            -> __post._paused
66     */
67    function pause() public onlyPauser whenNotPaused {
68      _paused = true;
69      emit Paused(msg.sender);
70    }
71
72    /**
73     * @dev called by the owner to unpause, returns to normal state
74     */
75    //@CTK NO_OVERFLOW
76    //@CTK NO_BUF_OVERFLOW
77    //@CTK NO_ASF
78    /*@CTK "Pausable unpause correctness"
79      @post !_paused -> __reverted
80      @post msg.sender == 0x0 -> __reverted
81      @post !pausers.bearer[msg.sender] -> __reverted
82      @post _paused && msg.sender != 0x0 && pausers.bearer[msg.sender]
83            -> !__post._paused
84     */
85    function unpause() public onlyPauser whenPaused {
86      _paused = false;
87      emit Unpaused(msg.sender);
88    }
89  }
```

File openzeppelin-solidity/contracts/math/SafeMath.sol

```
1  pragma solidity ^0.4.24;
2
3  /**
4   * @title SafeMath
5   * @dev Math operations with safety checks that revert on error
6   */
7  library SafeMath {
8
9    /**
10   * @dev Multiplies two numbers, reverts on overflow.
11   */
12   /*@CTK "SafeMath mul"
13     @post (((a) > (0)) && ((((a) * (b)) / (a)) != (b))) == (__reverted)
14     @post !__reverted -> __return == a * b
15     @post !__reverted == !__has_overflow
16     @post !(__has_buf_overflow)
17     @post !(__has_assertion_failure)
18     */
19   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
20     // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
21     // benefit is lost if 'b' is also tested.
22     // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
23     if (a == 0) {
24       return 0;
25     }
26
27     uint256 c = a * b;
28     require(c / a == b);
29
30     return c;
31   }
32
33   /**
34   * @dev Integer division of two numbers truncating the quotient, reverts on division
       by zero.
35   */
36   /*@CTK "SafeMath div"
37     @post b != 0 -> !__reverted
38     @post !__reverted -> __return == a / b
39     @post !__reverted -> !__has_overflow
40     @post !(__has_buf_overflow)
41     @post !(__has_assertion_failure)
42   */
43   function div(uint256 a, uint256 b) internal pure returns (uint256) {
44     require(b > 0); // Solidity only automatically asserts when dividing by 0
45     uint256 c = a / b;
46     // assert(a == b * c + a % b); // There is no case in which this doesn't hold
47
48     return c;
49   }
50
51   /**
52   * @dev Subtracts two numbers, reverts on overflow (i.e. if subtrahend is greater
       than minuend).
53   */
54   /*@CTK "SafeMath sub"
55     @post (a < b) == __reverted
56     @post !__reverted -> __return == a - b
```

```
    @post !__reverted -> !__has_overflow
    @post !(__has_buf_overflow)
    @post !(__has_assertion_failure)
  */
  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    require(b <= a);
    uint256 c = a - b;

    return c;
  }

  /**
  * @dev Adds two numbers, reverts on overflow.
  */
  /*@CTK "SafeMath add"
    @post (a + b < a || a + b < b) == __reverted
    @post !__reverted -> __return == a + b
    @post !__reverted -> !__has_overflow
    @post !(__has_buf_overflow)
    @post !(__has_assertion_failure)
  */
  function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    require(c >= a);

    return c;
  }

  /**
  * @dev Divides two numbers and returns the remainder (unsigned integer modulo),
  * reverts when dividing by zero.
  */
  /*@CTK "SafeMath div"
    @post b != 0 -> !__reverted
    @post !__reverted -> __return == a % b
    @post !__reverted -> !__has_overflow
    @post !(__has_buf_overflow)
    @post !(__has_assertion_failure)
  */
  function mod(uint256 a, uint256 b) internal pure returns (uint256) {
    require(b != 0);
    return a % b;
  }
}
```

File openzeppelin-solidity/contracts/token/ERC20/ERC20.sol

```
pragma solidity ^0.4.24;

import "./IERC20.sol";
import "../../math/SafeMath.sol";

/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
 * Originally based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/
    master/smart_contract/FirstBloodToken.sol
```

```
12  */
13 contract ERC20 is IERC20 {
14   using SafeMath for uint256;
15
16   mapping (address => uint256) private _balances;
17
18   mapping (address => mapping (address => uint256)) private _allowed;
19
20   uint256 private _totalSupply;
21
22   /**
23   * @dev Total number of tokens in existence
24   */
25   //@CTK NO_OVERFLOW
26   //@CTK NO_BUF_OVERFLOW
27   //@CTK NO_ASF
28   /*@CTK "totalSupply correctness"
29     @post __return == _totalSupply
30    */
31   function totalSupply() public view returns (uint256) {
32     return _totalSupply;
33   }
34
35   /**
36   * @dev Gets the balance of the specified address.
37   * @param owner The address to query the balance of.
38   * @return An uint256 representing the amount owned by the passed address.
39   */
40   //@CTK NO_OVERFLOW
41   //@CTK NO_BUF_OVERFLOW
42   //@CTK NO_ASF
43   /*@CTK "balanceOf correctness"
44     @post __return == _balances[owner]
45    */
46   function balanceOf(address owner) public view returns (uint256) {
47     return _balances[owner];
48   }
49
50   /**
51    * @dev Function to check the amount of tokens that an owner allowed to a spender.
52    * @param owner address The address which owns the funds.
53    * @param spender address The address which will spend the funds.
54    * @return A uint256 specifying the amount of tokens still available for the spender
         .
55    */
56   //@CTK NO_OVERFLOW
57   //@CTK NO_BUF_OVERFLOW
58   //@CTK NO_ASF
59   /*@CTK "allowance correctness"
60     @post __return == _allowed[owner][spender]
61    */
62   function allowance(
63     address owner,
64     address spender
65    )
66     public
67     view
68     returns (uint256)
```

```
69  {
70    return _allowed[owner][spender];
71  }
72
73  /**
74  * @dev Transfer token for a specified address
75  * @param to The address to transfer to.
76  * @param value The amount to be transferred.
77  */
78  //@CTK NO_OVERFLOW
79  //@CTK NO_BUF_OVERFLOW
80  //@CTK NO_ASF
81  /*@CTK "transfer correctness"
82    @tag assume_completion
83    @post to != 0x0
84    @post value <= _balances[msg.sender]
85    @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
          value
86    @post to != msg.sender -> __post._balances[to] == _balances[to] + value
87    @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
88   */
89  function transfer(address to, uint256 value) public returns (bool) {
90    _transfer(msg.sender, to, value);
91    return true;
92  }
93
94  /**
95   * @dev Approve the passed address to spend the specified amount of tokens on behalf
          of msg.sender.
96   * Beware that changing an allowance with this method brings the risk that someone
         may use both the old
97   * and the new allowance by unfortunate transaction ordering. One possible solution
         to mitigate this
98   * race condition is to first reduce the spender's allowance to 0 and set the
         desired value afterwards:
99   * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
100  * @param spender The address which will spend the funds.
101  * @param value The amount of tokens to be spent.
102  */
103 //@CTK NO_OVERFLOW
104 //@CTK NO_BUF_OVERFLOW
105 //@CTK NO_ASF
106 /*@CTK "approve correctness"
107   @post spender == 0x0 -> __reverted
108   @post spender != 0x0 -> __post._allowed[msg.sender][spender] == value
109 */
110 function approve(address spender, uint256 value) public returns (bool) {
111   require(spender != address(0));
112
113   _allowed[msg.sender][spender] = value;
114   emit Approval(msg.sender, spender, value);
115   return true;
116 }
117
118 /**
119  * @dev Transfer tokens from one address to another
120  * @param from address The address which you want to send tokens from
121  * @param to address The address which you want to transfer to
```

```
122   * @param value uint256 the amount of tokens to be transferred
123   */
124  //@CTK NO_OVERFLOW
125  //@CTK NO_BUF_OVERFLOW
126  //@CTK NO_ASF
127  /*@CTK "transferFrom correctness"
128    @tag assume_completion
129    @post to != 0x0
130    @post value <= _balances[from] && value <= _allowed[from][msg.sender]
131    @post to != from -> __post._balances[from] == _balances[from] - value
132    @post to != from -> __post._balances[to] == _balances[to] + value
133    @post to == from -> __post._balances[from] == _balances[from]
134    @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
135   */
136  function transferFrom(
137    address from,
138    address to,
139    uint256 value
140  )
141    public
142    returns (bool)
143  {
144    require(value <= _allowed[from][msg.sender]);
145
146    _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
147    _transfer(from, to, value);
148    return true;
149  }
150
151  /**
152   * @dev Increase the amount of tokens that an owner allowed to a spender.
153   * approve should be called when allowed_[_spender] == 0. To increment
154   * allowed value is better to use this function to avoid 2 calls (and wait until
155   * the first transaction is mined)
156   * From MonolithDAO Token.sol
157   * @param spender The address which will spend the funds.
158   * @param addedValue The amount of tokens to increase the allowance by.
159   */
160  //@CTK NO_OVERFLOW
161  //@CTK NO_BUF_OVERFLOW
162  //@CTK NO_ASF
163  /*@CTK "increaseAllowance correctness"
164    @tag assume_completion
165    @post spender != 0x0
166    @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
         addedValue
167  */
168  function increaseAllowance(
169    address spender,
170    uint256 addedValue
171  )
172    public
173    returns (bool)
174  {
175    require(spender != address(0));
176
177    _allowed[msg.sender][spender] = (
178      _allowed[msg.sender][spender].add(addedValue));
```

```
179     emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
180     return true;
181   }
182
183   /**
184    * @dev Decrease the amount of tokens that an owner allowed to a spender.
185    * approve should be called when allowed_[_spender] == 0. To decrement
186    * allowed value is better to use this function to avoid 2 calls (and wait until
187    * the first transaction is mined)
188    * From MonolithDAO Token.sol
189    * @param spender The address which will spend the funds.
190    * @param subtractedValue The amount of tokens to decrease the allowance by.
191    */
192   //@CTK NO_OVERFLOW
193   //@CTK NO_BUF_OVERFLOW
194   //@CTK NO_ASF
195   /*@CTK "decreaseAllowance correctness"
196     @tag assume_completion
197     @post spender != 0x0
198     @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
          subtractedValue
199   */
200   function decreaseAllowance(
201     address spender,
202     uint256 subtractedValue
203   )
204     public
205     returns (bool)
206   {
207     require(spender != address(0));
208
209     _allowed[msg.sender][spender] = (
210       _allowed[msg.sender][spender].sub(subtractedValue));
211     emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
212     return true;
213   }
214
215   /**
216   * @dev Transfer token for a specified addresses
217   * @param from The address to transfer from.
218   * @param to The address to transfer to.
219   * @param value The amount to be transferred.
220   */
221   //@CTK NO_OVERFLOW
222   //@CTK NO_BUF_OVERFLOW
223   //@CTK NO_ASF
224   /*@CTK "_transfer correctness"
225     @tag assume_completion
226     @post to != 0x0
227     @post value <= _balances[from]
228     @post to != from -> __post._balances[from] == _balances[from] - value
229     @post to != from -> __post._balances[to] == _balances[to] + value
230     @post to == from -> __post._balances[from] == _balances[from]
231    */
232   function _transfer(address from, address to, uint256 value) internal {
233     require(value <= _balances[from]);
234     require(to != address(0));
235
```

```
236     _balances[from] = _balances[from].sub(value);
237     _balances[to] = _balances[to].add(value);
238     emit Transfer(from, to, value);
239   }
240
241   /**
242    * @dev Internal function that mints an amount of the token and assigns it to
243    * an account. This encapsulates the modification of balances such that the
244    * proper events are emitted.
245    * @param account The account that will receive the created tokens.
246    * @param value The amount that will be created.
247    */
248   //@CTK NO_OVERFLOW
249   //@CTK NO_BUF_OVERFLOW
250   //@CTK NO_ASF
251   /*@CTK "_mint correctness"
252     @tag assume_completion
253     @post account != 0x0
254     @post __post._balances[account] == _balances[account] + value
255     @post __post._totalSupply == _totalSupply + value
256    */
257   function _mint(address account, uint256 value) internal {
258     require(account != 0);
259     _totalSupply = _totalSupply.add(value);
260     _balances[account] = _balances[account].add(value);
261     emit Transfer(address(0), account, value);
262   }
263
264   /**
265    * @dev Internal function that burns an amount of the token of a given
266    * account.
267    * @param account The account whose tokens will be burnt.
268    * @param value The amount that will be burnt.
269    */
270   //@CTK NO_OVERFLOW
271   //@CTK NO_BUF_OVERFLOW
272   //@CTK NO_ASF
273   /*@CTK "_burn correctness"
274     @tag assume_completion
275     @post account != 0x0
276     @post value <= _balances[account]
277     @post __post._balances[account] == _balances[account] - value
278     @post __post._totalSupply == _totalSupply - value
279    */
280   function _burn(address account, uint256 value) internal {
281     require(account != 0);
282     require(value <= _balances[account]);
283
284     _totalSupply = _totalSupply.sub(value);
285     _balances[account] = _balances[account].sub(value);
286     emit Transfer(account, address(0), value);
287   }
288
289   /**
290    * @dev Internal function that burns an amount of the token of a given
291    * account, deducting from the sender's allowance for said account. Uses the
292    * internal burn function.
293    * @param account The account whose tokens will be burnt.
```

```
294     * @param value The amount that will be burnt.
295     */
296   //@CTK NO_OVERFLOW
297   //@CTK NO_BUF_OVERFLOW
298   //@CTK NO_ASF
299   /*@CTK "_burnFrom correctness"
300     @tag assume_completion
301     @post account != 0x0
302     @post value <= _balances[account] && value <= _allowed[account][msg.sender]
303     @post __post._balances[account] == _balances[account] - value
304     @post __post._totalSupply == _totalSupply - value
305     @post __post._allowed[account][msg.sender] == _allowed[account][msg.sender] -
          value
306    */
307   function _burnFrom(address account, uint256 value) internal {
308     require(value <= _allowed[account][msg.sender]);
309
310     // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
          ,
311     // this function needs to emit an event with the updated approval.
312     _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
313       value);
314     _burn(account, value);
315   }
316 }
```

File openzeppelin-solidity/contracts/token/ERC20/ERC20Detailed.sol

```
1  pragma solidity ^0.4.24;
2
3  import "./IERC20.sol";
4
5  /**
6   * @title ERC20Detailed token
7   * @dev The decimals are only for visualization purposes.
8   * All the operations are done using the smallest and indivisible token unit,
9   * just as on Ethereum all the operations are done in wei.
10  */
11 contract ERC20Detailed is IERC20 {
12   string private _name;
13   string private _symbol;
14   uint8 private _decimals;
15
16   //@CTK NO_OVERFLOW
17   //@CTK NO_BUF_OVERFLOW
18   //@CTK NO_ASF
19   /*@CTK "ERC20Detailed constructor correctness"
20     @post __post._name == name
21     @post __post._symbol == symbol
22     @post __post._decimals == decimals
23    */
24   constructor(string name, string symbol, uint8 decimals) public {
25     _name = name;
26     _symbol = symbol;
27     _decimals = decimals;
28   }
29
30   /**
31    * @return the name of the token.
```

```
32     */
33   //@CTK NO_OVERFLOW
34   //@CTK NO_BUF_OVERFLOW
35   //@CTK NO_ASF
36   /*@CTK "ERC20Detailed name correctness"
37     @post __return == _name
38    */
39   function name() public view returns(string) {
40     return _name;
41   }
42
43   /**
44    * @return the symbol of the token.
45    */
46   //@CTK NO_OVERFLOW
47   //@CTK NO_BUF_OVERFLOW
48   //@CTK NO_ASF
49   /*@CTK "ERC20Detailed symbol correctness"
50     @post __return == _symbol
51    */
52   function symbol() public view returns(string) {
53     return _symbol;
54   }
55
56   //@CTK NO_OVERFLOW
57   //@CTK NO_BUF_OVERFLOW
58   //@CTK NO_ASF
59   /*@CTK "ERC20Detailed decimals correctness"
60     @post __return == _decimals
61    */
62   /**
63    * @return the number of decimals of the token.
64    */
65   function decimals() public view returns(uint8) {
66     return _decimals;
67   }
68 }
```

File openzeppelin-solidity/contracts/token/ERC20/ERC20Pausable.sol

```
1  pragma solidity ^0.4.24;
2
3  import "./ERC20.sol";
4  import "../../lifecycle/Pausable.sol";
5
6  /**
7   * @title Pausable token
8   * @dev ERC20 modified with pausable transfers.
9   **/
10 contract ERC20Pausable is ERC20, Pausable {
11
12   //@CTK NO_OVERFLOW
13   //@CTK NO_BUF_OVERFLOW
14   //@CTK NO_ASF
15   /*@CTK "ERC20Pausable transfer correctness"
16     @tag assume_completion
17     @post to != 0x0
18     @post value <= _balances[msg.sender]
19     @post _paused == false
```

```
20    @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
          value
21    @post to != msg.sender -> __post._balances[to] == _balances[to] + value
22    @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
23   */
24  function transfer(
25    address to,
26    uint256 value
27  )
28    public
29    whenNotPaused
30    returns (bool)
31  {
32    return super.transfer(to, value);
33  }
34
35  //@CTK NO_OVERFLOW
36  //@CTK NO_BUF_OVERFLOW
37  //@CTK NO_ASF
38  /*@CTK "ERC20Pausable transferFrom correctness"
39    @tag assume_completion
40    @post to != 0x0
41    @post value <= _balances[from] && value <= _allowed[from][msg.sender]
42    @post _paused == false
43    @post to != from -> __post._balances[from] == _balances[from] - value
44    @post to != from -> __post._balances[to] == _balances[to] + value
45    @post to == from -> __post._balances[from] == _balances[from]
46    @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
47   */
48  function transferFrom(
49    address from,
50    address to,
51    uint256 value
52  )
53    public
54    whenNotPaused
55    returns (bool)
56  {
57    return super.transferFrom(from, to, value);
58  }
59
60  //@CTK NO_OVERFLOW
61  //@CTK NO_BUF_OVERFLOW
62  //@CTK NO_ASF
63  /*@CTK "ERC20Pausable approve correctness"
64    @post spender == 0x0 -> __reverted
65    @post _paused -> __reverted
66    @post spender != 0x0 && !_paused
67          -> __post._allowed[msg.sender][spender] == value
68  */
69  function approve(
70    address spender,
71    uint256 value
72  )
73    public
74    whenNotPaused
75    returns (bool)
76  {
```

```
77      return super.approve(spender, value);
78    }
79
80    //@CTK NO_OVERFLOW
81    //@CTK NO_BUF_OVERFLOW
82    //@CTK NO_ASF
83    /*@CTK "ERC20Pausable increaseAllowance correctness"
84      @tag assume_completion
85      @post spender != 0x0
86      @post _paused == false
87      @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
            addedValue
88    */
89    function increaseAllowance(
90      address spender,
91      uint addedValue
92    )
93      public
94      whenNotPaused
95      returns (bool success)
96    {
97      return super.increaseAllowance(spender, addedValue);
98    }
99
100   //@CTK NO_OVERFLOW
101   //@CTK NO_BUF_OVERFLOW
102   //@CTK NO_ASF
103   /*@CTK "ERC20Pausable decreaseAllowance correctness"
104     @tag assume_completion
105     @post spender != 0x0
106     @post _paused == false
107     @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
            subtractedValue
108   */
109   function decreaseAllowance(
110     address spender,
111     uint subtractedValue
112   )
113     public
114     whenNotPaused
115     returns (bool success)
116   {
117     return super.decreaseAllowance(spender, subtractedValue);
118   }
119 }
```

File openzeppelin-solidity/contracts/token/ERC20/SafeERC20.sol

```
1  pragma solidity ^0.4.24;
2
3  import "./IERC20.sol";
4  import "../../math/SafeMath.sol";
5
6  /**
7   * @title SafeERC20
8   * @dev Wrappers around ERC20 operations that throw on failure.
9   * To use this library you can add a `using SafeERC20 for ERC20;` statement to your
        contract,
10  * which allows you to call the safe operations as `token.safeTransfer(...)`, etc.
```

```
 */
library SafeERC20 {

  using SafeMath for uint256;

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  function safeTransfer(
    IERC20 token,
    address to,
    uint256 value
  )
    internal
  {
    require(token.transfer(to, value));
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  function safeTransferFrom(
    IERC20 token,
    address from,
    address to,
    uint256 value
  )
    internal
  {
    require(token.transferFrom(from, to, value));
  }

  //@CTK NO_OVERFLOW
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_ASF
  function safeApprove(
    IERC20 token,
    address spender,
    uint256 value
  )
    internal
  {
    // safeApprove should only be called when setting an initial allowance,
    // or when resetting it to zero. To increase and decrease it, use
    // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
    require((value == 0) || (token.allowance(msg.sender, spender) == 0));
    require(token.approve(spender, value));
  }

  function safeIncreaseAllowance(
    IERC20 token,
    address spender,
    uint256 value
  )
    internal
  {
    uint256 newAllowance = token.allowance(address(this), spender).add(value);
    require(token.approve(spender, newAllowance));
```

```
69    }
70
71    function safeDecreaseAllowance(
72      IERC20 token,
73      address spender,
74      uint256 value
75    )
76      internal
77    {
78      uint256 newAllowance = token.allowance(address(this), spender).sub(value);
79      require(token.approve(spender, newAllowance));
80    }
81  }
```

# 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 chiliZ.sol

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

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

### INSECURE_COMPILER_VERSION

Line 1 in File Roles.sol

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

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

### INSECURE_COMPILER_VERSION

Line 1 in File PauserRole.sol

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

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

### INSECURE_COMPILER_VERSION

Line 1 in File Pausable.sol

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

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

### INSECURE_COMPILER_VERSION

Line 1 in File SafeMath.sol

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

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

### INSECURE_COMPILER_VERSION

Line 1 in File ERC20.sol

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

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

### INSECURE_COMPILER_VERSION

Line 1 in File ERC20Detailed.sol

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

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

### INSECURE_COMPILER_VERSION

Line 1 in File ERC20Pausable.sol

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

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

### INSECURE_COMPILER_VERSION

Line 1 in File SafeERC20.sol

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

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

# Formal Verification Request 1

**If method completes, integer overflow would not happen.**

22, Nov 2018
20.56 ms

Line 15 in File Roles.sol

```
15  //@CTK NO_OVERFLOW
```

Line 23-28 in File Roles.sol

```
23  function add(Role storage role, address account) internal {
24    require(account != address(0));
25    require(!role.bearer[account]);
26
27    role.bearer[account] = true;
28  }
```

The code meets the specification

# Formal Verification Request 2

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

22, Nov 2018
0.55 ms

Line 16 in File Roles.sol

```
16  //@CTK NO_BUF_OVERFLOW
```

Line 23-28 in File Roles.sol

```
23  function add(Role storage role, address account) internal {
24    require(account != address(0));
25    require(!role.bearer[account]);
26
27    role.bearer[account] = true;
28  }
```

The code meets the specification

# Formal Verification Request 3

**Method will not encounter an assertion failure.**

22, Nov 2018
0.6 ms

Line 17 in File Roles.sol

```
17  //@CTK NO_ASF
```

Line 23-28 in File Roles.sol

```
23  function add(Role storage role, address account) internal {
24    require(account != address(0));
25    require(!role.bearer[account]);
26
27    role.bearer[account] = true;
28  }
```

The code meets the specification

# Formal Verification Request 4

**Roles add correctness**

22, Nov 2018
2.16 ms

Line 18-22 in File Roles.sol

```
18  /*@CTK "Roles add correctness"
19    @post account == 0x0 -> __reverted
20    @post role.bearer[account] -> __reverted
21    @post account != 0x0 && !role.bearer[account] -> !__reverted
22   */
```

Line 23-28 in File Roles.sol

```
23  function add(Role storage role, address account) internal {
24    require(account != address(0));
25    require(!role.bearer[account]);
26
27    role.bearer[account] = true;
28  }
```

The code meets the specification

# Formal Verification Request 5

**If method completes, integer overflow would not happen.**

22, Nov 2018
22.33 ms

Line 33 in File Roles.sol

```
33  //@CTK NO_OVERFLOW
```

Line 41-46 in File Roles.sol

```
41  function remove(Role storage role, address account) internal {
42    require(account != address(0));
43    require(role.bearer[account]);
44
45    role.bearer[account] = false;
46  }
```

The code meets the specification

# Formal Verification Request 6

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

📅 22, Nov 2018
⏱ 0.59 ms

Line 34 in File Roles.sol

```
34  //@CTK NO_BUF_OVERFLOW
```

Line 41-46 in File Roles.sol

```
41  function remove(Role storage role, address account) internal {
42    require(account != address(0));
43    require(role.bearer[account]);
44
45    role.bearer[account] = false;
46  }
```

✅ The code meets the specification

# Formal Verification Request 7

**Method will not encounter an assertion failure.**

📅 22, Nov 2018
⏱ 0.57 ms

Line 35 in File Roles.sol

```
35  //@CTK NO_ASF
```

Line 41-46 in File Roles.sol

```
41  function remove(Role storage role, address account) internal {
42    require(account != address(0));
43    require(role.bearer[account]);
44
45    role.bearer[account] = false;
46  }
```

✅ The code meets the specification

# Formal Verification Request 8

**Roles add correctness**

📅 22, Nov 2018
⏱ 1.67 ms

Line 36-40 in File Roles.sol

```
36  /*@CTK "Roles add correctness"
37    @post account == 0x0 -> __reverted
38    @post !role.bearer[account] -> __reverted
39    @post account != 0x0 && role.bearer[account] -> !__reverted
40   */
```

Line 41-46 in File Roles.sol

```
41  function remove(Role storage role, address account) internal {
42    require(account != address(0));
43    require(role.bearer[account]);
44
45    role.bearer[account] = false;
46  }
```

The code meets the specification

# Formal Verification Request 9

**If method completes, integer overflow would not happen.**

22, Nov 2018
16.66 ms

Line 52 in File Roles.sol

```
52  //@CTK NO_OVERFLOW
```

Line 59-66 in File Roles.sol

```
59  function has(Role storage role, address account)
60    internal
61    view
62    returns (bool)
63  {
64    require(account != address(0));
65    return role.bearer[account];
66  }
```

The code meets the specification

# Formal Verification Request 10

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

22, Nov 2018
0.49 ms

Line 53 in File Roles.sol

```
53  //@CTK NO_BUF_OVERFLOW
```

Line 59-66 in File Roles.sol

```
59  function has(Role storage role, address account)
60    internal
61    view
62    returns (bool)
63  {
64    require(account != address(0));
65    return role.bearer[account];
66  }
```

The code meets the specification

# Formal Verification Request 11

**Method will not encounter an assertion failure.**

22, Nov 2018
0.53 ms

Line 54 in File Roles.sol

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

Line 59-66 in File Roles.sol

```
59  function has(Role storage role, address account)
60    internal
61    view
62    returns (bool)
63  {
64    require(account != address(0));
65    return role.bearer[account];
66  }
```

The code meets the specification

# Formal Verification Request 12

**Roles has correctness**

22, Nov 2018
1.19 ms

Line 55-58 in File Roles.sol

```
55  /*@CTK "Roles has correctness"
56    @post account == 0x0 -> __reverted
57    @post account != 0x0 -> (!__reverted) && (__return == role.bearer[account])
58   */
```

Line 59-66 in File Roles.sol

```
59  function has(Role storage role, address account)
60    internal
61    view
62    returns (bool)
```

```
63  {
64    require(account != address(0));
65    return role.bearer[account];
66  }
```

The code meets the specification

## Formal Verification Request 13

**If method completes, integer overflow would not happen.**

22, Nov 2018
114.41 ms

Line 13 in File PauserRole.sol

```
13  //@CTK NO_OVERFLOW
```

Line 22-24 in File PauserRole.sol

```
22  constructor() internal {
23    _addPauser(msg.sender);
24  }
```

The code meets the specification

## Formal Verification Request 14

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

22, Nov 2018
0.83 ms

Line 14 in File PauserRole.sol

```
14  //@CTK NO_BUF_OVERFLOW
```

Line 22-24 in File PauserRole.sol

```
22  constructor() internal {
23    _addPauser(msg.sender);
24  }
```

The code meets the specification

## Formal Verification Request 15

**Method will not encounter an assertion failure.**

22, Nov 2018
0.84 ms

Line 15 in File PauserRole.sol

```
15  //@CTK NO_ASF
```

Line 22-24 in File PauserRole.sol

```
22  constructor() internal {
23    _addPauser(msg.sender);
24  }
```

The code meets the specification

# Formal Verification Request 16

**PauserRole constructor correctness**

22, Nov 2018
2.9 ms

Line 16-21 in File PauserRole.sol

```
16  /*@CTK "PauserRole constructor correctness"
17    @post msg.sender == 0x0 -> __reverted
18    @post pausers.bearer[msg.sender] -> __reverted
19    @post msg.sender != 0x0 && !pausers.bearer[msg.sender]
20          -> !__reverted && __post.pausers.bearer[msg.sender]
21   */
```

Line 22-24 in File PauserRole.sol

```
22  constructor() internal {
23    _addPauser(msg.sender);
24  }
```

The code meets the specification

# Formal Verification Request 17

**If method completes, integer overflow would not happen.**

22, Nov 2018
53.61 ms

Line 31 in File PauserRole.sol

```
31  //@CTK NO_OVERFLOW
```

Line 38-40 in File PauserRole.sol

```
38  function isPauser(address account) public view returns (bool) {
39    return pausers.has(account);
40  }
```

The code meets the specification

# Formal Verification Request 18

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

22, Nov 2018
0.98 ms

Line 32 in File PauserRole.sol

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

Line 38-40 in File PauserRole.sol

```
38  function isPauser(address account) public view returns (bool) {
39    return pausers.has(account);
40  }
```

The code meets the specification

# Formal Verification Request 19

**Method will not encounter an assertion failure.**

22, Nov 2018
0.72 ms

Line 33 in File PauserRole.sol

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

Line 38-40 in File PauserRole.sol

```
38  function isPauser(address account) public view returns (bool) {
39    return pausers.has(account);
40  }
```

The code meets the specification

# Formal Verification Request 20

**isBurner correctness**

22, Nov 2018
1.7 ms

Line 34-37 in File PauserRole.sol

```
34  /*@CTK "isBurner correctness"
35    @post account == 0x0 -> __reverted
36    @post account != 0x0 -> !__reverted && __return == pausers.bearer[account]
37   */
```

Line 38-40 in File PauserRole.sol

```
38  function isPauser(address account) public view returns (bool) {
39    return pausers.has(account);
40  }
```

The code meets the specification

## Formal Verification Request 21

**If method completes, integer overflow would not happen.**

22, Nov 2018
108.36 ms

Line 42 in File PauserRole.sol

```
42  //@CTK NO_OVERFLOW
```

Line 54-56 in File PauserRole.sol

```
54  function addPauser(address account) public onlyPauser {
55    _addPauser(account);
56  }
```

The code meets the specification

## Formal Verification Request 22

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

22, Nov 2018
1.62 ms

Line 43 in File PauserRole.sol

```
43  //@CTK NO_BUF_OVERFLOW
```

Line 54-56 in File PauserRole.sol

```
54  function addPauser(address account) public onlyPauser {
55    _addPauser(account);
56  }
```

The code meets the specification

## Formal Verification Request 23

**Method will not encounter an assertion failure.**

22, Nov 2018
1.51 ms

Line 44 in File PauserRole.sol

```
44  //@CTK NO_ASF
```

Line 54-56 in File PauserRole.sol

```
54  function addPauser(address account) public onlyPauser {
55    _addPauser(account);
56  }
```

The code meets the specification

# Formal Verification Request 24

**_addPauser correctness**

22, Nov 2018
4.67 ms

Line 45-53 in File PauserRole.sol

```
45  /*@CTK "_addPauser correctness"
46    @post account == 0x0 -> __reverted
47    @post msg.sender == 0x0 -> __reverted
48    @post pausers.bearer[account] -> __reverted
49    @post !pausers.bearer[msg.sender] -> __reverted
50    @post account != 0x0 && !pausers.bearer[account]
51          && msg.sender != 0x0 && pausers.bearer[msg.sender]
52          -> !__reverted && __post.pausers.bearer[account]
53   */
```

Line 54-56 in File PauserRole.sol

```
54  function addPauser(address account) public onlyPauser {
55    _addPauser(account);
56  }
```

The code meets the specification

# Formal Verification Request 25

**If method completes, integer overflow would not happen.**

22, Nov 2018
112.25 ms

Line 58 in File PauserRole.sol

```
58  //@CTK NO_OVERFLOW
```

Line 67-69 in File PauserRole.sol

```
67  function renouncePauser() public {
68    _removePauser(msg.sender);
69  }
```

The code meets the specification

# Formal Verification Request 26

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

22, Nov 2018
0.8 ms

Line 59 in File PauserRole.sol

```
59  //@CTK NO_BUF_OVERFLOW
```

Line 67-69 in File PauserRole.sol

```
67  function renouncePauser() public {
68    _removePauser(msg.sender);
69  }
```

The code meets the specification

# Formal Verification Request 27

**Method will not encounter an assertion failure.**

22, Nov 2018
0.79 ms

Line 60 in File PauserRole.sol

```
60  //@CTK NO_ASF
```

Line 67-69 in File PauserRole.sol

```
67  function renouncePauser() public {
68    _removePauser(msg.sender);
69  }
```

The code meets the specification

# Formal Verification Request 28

**renouncePauser correctness**

22, Nov 2018
2.83 ms

Line 61-66 in File PauserRole.sol

```
61  /*@CTK "renouncePauser correctness"
62    @post msg.sender == 0x0 -> __reverted
63    @post !pausers.bearer[msg.sender] -> __reverted
64    @post msg.sender != 0x0 && pausers.bearer[msg.sender]
65          -> !__reverted && !__post.pausers.bearer[msg.sender]
66   */
```

Line 67-69 in File PauserRole.sol

```
67  function renouncePauser() public {
68    _removePauser(msg.sender);
69  }
```

The code meets the specification

## Formal Verification Request 29

**If method completes, integer overflow would not happen.**

22, Nov 2018
0.76 ms

Line 71 in File PauserRole.sol

```
71  //@CTK NO_OVERFLOW
```

Line 80-83 in File PauserRole.sol

```
80  function _addPauser(address account) internal {
81    pausers.add(account);
82    emit PauserAdded(account);
83  }
```

The code meets the specification

## Formal Verification Request 30

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

22, Nov 2018
0.66 ms

Line 72 in File PauserRole.sol

```
72  //@CTK NO_BUF_OVERFLOW
```

Line 80-83 in File PauserRole.sol

```
80  function _addPauser(address account) internal {
81    pausers.add(account);
82    emit PauserAdded(account);
83  }
```

The code meets the specification

## Formal Verification Request 31

**Method will not encounter an assertion failure.**

22, Nov 2018
0.66 ms

Line 73 in File PauserRole.sol

```
73  //@CTK NO_ASF
```

Line 80-83 in File PauserRole.sol

```
80  function _addPauser(address account) internal {
81    pausers.add(account);
82    emit PauserAdded(account);
83  }
```

The code meets the specification

# Formal Verification Request 32

**_addPauser correctness**

22, Nov 2018
2.25 ms

Line 74-79 in File PauserRole.sol

```
74  /*@CTK "_addPauser correctness"
75    @post account == 0x0 -> __reverted
76    @post pausers.bearer[account] -> __reverted
77    @post account != 0x0 && !pausers.bearer[account]
78          -> !__reverted && __post.pausers.bearer[account]
79   */
```

Line 80-83 in File PauserRole.sol

```
80  function _addPauser(address account) internal {
81    pausers.add(account);
82    emit PauserAdded(account);
83  }
```

The code meets the specification

# Formal Verification Request 33

**If method completes, integer overflow would not happen.**

22, Nov 2018
0.67 ms

Line 85 in File PauserRole.sol

```
85  //@CTK NO_OVERFLOW
```

Line 94-97 in File PauserRole.sol

```
94  function _removePauser(address account) internal {
95    pausers.remove(account);
96    emit PauserRemoved(account);
97  }
```

The code meets the specification

# Formal Verification Request 34

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

22, Nov 2018
0.71 ms

Line 86 in File PauserRole.sol

```
86  //@CTK NO_BUF_OVERFLOW
```

Line 94-97 in File PauserRole.sol

```
94  function _removePauser(address account) internal {
95    pausers.remove(account);
96    emit PauserRemoved(account);
97  }
```

The code meets the specification

# Formal Verification Request 35

**Method will not encounter an assertion failure.**

22, Nov 2018
0.65 ms

Line 87 in File PauserRole.sol

```
87  //@CTK NO_ASF
```

Line 94-97 in File PauserRole.sol

```
94  function _removePauser(address account) internal {
95    pausers.remove(account);
96    emit PauserRemoved(account);
97  }
```

The code meets the specification

# Formal Verification Request 36

**_removePauser correctness**

22, Nov 2018
2.45 ms

Line 88-93 in File PauserRole.sol

```
88  /*@CTK "_removePauser correctness"
89    @post account == 0x0 -> __reverted
90    @post !pausers.bearer[account] -> __reverted
91    @post account != 0x0 && pausers.bearer[account]
92          -> !__reverted && !__post.pausers.bearer[account]
93   */
```

Line 94-97 in File PauserRole.sol

```
94  function _removePauser(address account) internal {
95    pausers.remove(account);
96    emit PauserRemoved(account);
97  }
```

The code meets the specification

## Formal Verification Request 37

**If method completes, integer overflow would not happen.**

22, Nov 2018
5.81 ms

Line 15 in File Pausable.sol

```
15  //@CTK NO_OVERFLOW
```

Line 21-23 in File Pausable.sol

```
21  constructor() internal {
22    _paused = false;
23  }
```

The code meets the specification

## Formal Verification Request 38

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

22, Nov 2018
0.47 ms

Line 16 in File Pausable.sol

```
16  //@CTK NO_BUF_OVERFLOW
```

Line 21-23 in File Pausable.sol

```
21  constructor() internal {
22    _paused = false;
23  }
```

The code meets the specification

## Formal Verification Request 39

**Method will not encounter an assertion failure.**

22, Nov 2018
0.44 ms

Line 17 in File Pausable.sol

```
17  //@CTK NO_ASF
```

Line 21-23 in File Pausable.sol

```
21  constructor() internal {
22    _paused = false;
23  }
```

The code meets the specification

# Formal Verification Request 40

**Pausable constructor correctness**

22, Nov 2018
0.8 ms

Line 18-20 in File Pausable.sol

```
18  /*@CTK "Pausable constructor correctness"
19    @post __post._paused == false
20   */
```

Line 21-23 in File Pausable.sol

```
21  constructor() internal {
22    _paused = false;
23  }
```

The code meets the specification

# Formal Verification Request 41

**If method completes, integer overflow would not happen.**

22, Nov 2018
6.8 ms

Line 28 in File Pausable.sol

```
28  //@CTK NO_OVERFLOW
```

Line 34-36 in File Pausable.sol

```
34  function paused() public view returns(bool) {
35    return _paused;
36  }
```

The code meets the specification

# Formal Verification Request 42

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

📅 22, Nov 2018
⏱ 0.39 ms

Line 29 in File Pausable.sol

```
29  //@CTK NO_BUF_OVERFLOW
```

Line 34-36 in File Pausable.sol

```
34  function paused() public view returns(bool) {
35    return _paused;
36  }
```

✅ The code meets the specification

# Formal Verification Request 43

**Method will not encounter an assertion failure.**

📅 22, Nov 2018
⏱ 0.4 ms

Line 30 in File Pausable.sol

```
30  //@CTK NO_ASF
```

Line 34-36 in File Pausable.sol

```
34  function paused() public view returns(bool) {
35    return _paused;
36  }
```

✅ The code meets the specification

# Formal Verification Request 44

**Pausable paused correctness**

📅 22, Nov 2018
⏱ 0.43 ms

Line 31-33 in File Pausable.sol

```
31  /*@CTK "Pausable paused correctness"
32    @post __return == _paused
33   */
```

Line 34-36 in File Pausable.sol

```
34  function paused() public view returns(bool) {
35    return _paused;
36  }
```

✅ The code meets the specification

# Formal Verification Request 45

**If method completes, integer overflow would not happen.**

22, Nov 2018
131.04 ms

Line 57 in File Pausable.sol

```
57  //@CTK NO_OVERFLOW
```

Line 67-70 in File Pausable.sol

```
67  function pause() public onlyPauser whenNotPaused {
68    _paused = true;
69    emit Paused(msg.sender);
70  }
```

The code meets the specification

# Formal Verification Request 46

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

22, Nov 2018
1.04 ms

Line 58 in File Pausable.sol

```
58  //@CTK NO_BUF_OVERFLOW
```

Line 67-70 in File Pausable.sol

```
67  function pause() public onlyPauser whenNotPaused {
68    _paused = true;
69    emit Paused(msg.sender);
70  }
```

The code meets the specification

# Formal Verification Request 47

**Method will not encounter an assertion failure.**

22, Nov 2018
1.05 ms

Line 59 in File Pausable.sol

```
59  //@CTK NO_ASF
```

Line 67-70 in File Pausable.sol

```
67  function pause() public onlyPauser whenNotPaused {
68    _paused = true;
69    emit Paused(msg.sender);
70  }
```

The code meets the specification

# Formal Verification Request 48

**Pausable pause correctness**

22, Nov 2018
4.95 ms

Line 60-66 in File Pausable.sol

```
60  /*@CTK "Pausable pause correctness"
61    @post _paused -> __reverted
62    @post msg.sender == 0x0 -> __reverted
63    @post !pausers.bearer[msg.sender] -> __reverted
64    @post !_paused && msg.sender != 0x0 && pausers.bearer[msg.sender]
65          -> __post._paused
66   */
```

Line 67-70 in File Pausable.sol

```
67  function pause() public onlyPauser whenNotPaused {
68    _paused = true;
69    emit Paused(msg.sender);
70  }
```

The code meets the specification

# Formal Verification Request 49

**If method completes, integer overflow would not happen.**

22, Nov 2018
74.14 ms

Line 75 in File Pausable.sol

```
75  //@CTK NO_OVERFLOW
```

Line 85-88 in File Pausable.sol

```
85  function unpause() public onlyPauser whenPaused {
86    _paused = false;
87    emit Unpaused(msg.sender);
88  }
```

The code meets the specification

# Formal Verification Request 50

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

22, Nov 2018
1.14 ms

Line 76 in File Pausable.sol

```
76  //@CTK NO_BUF_OVERFLOW
```

Line 85-88 in File Pausable.sol

```
85  function unpause() public onlyPauser whenPaused {
86    _paused = false;
87    emit Unpaused(msg.sender);
88  }
```

The code meets the specification

# Formal Verification Request 51

**Method will not encounter an assertion failure.**

22, Nov 2018
1.26 ms

Line 77 in File Pausable.sol

```
77  //@CTK NO_ASF
```

Line 85-88 in File Pausable.sol

```
85  function unpause() public onlyPauser whenPaused {
86    _paused = false;
87    emit Unpaused(msg.sender);
88  }
```

The code meets the specification

# Formal Verification Request 52

**Pausable unpause correctness**

22, Nov 2018
4.29 ms

Line 78-84 in File Pausable.sol

```
78  /*@CTK "Pausable unpause correctness"
79    @post !_paused -> __reverted
80    @post msg.sender == 0x0 -> __reverted
81    @post !pausers.bearer[msg.sender] -> __reverted
82    @post _paused && msg.sender != 0x0 && pausers.bearer[msg.sender]
83         -> !__post._paused
84   */
```

Line 85-88 in File Pausable.sol

```
85  function unpause() public onlyPauser whenPaused {
86    _paused = false;
87    emit Unpaused(msg.sender);
88  }
```

The code meets the specification

# Formal Verification Request 53

### SafeMath mul

22, Nov 2018
444.38 ms

Line 12-18 in File SafeMath.sol

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

Line 19-31 in File SafeMath.sol

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

The code meets the specification

# Formal Verification Request 54

### SafeMath div

22, Nov 2018
16.56 ms

Line 36-42 in File SafeMath.sol

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

Line 43-49 in File SafeMath.sol

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

The code meets the specification

# Formal Verification Request 55

### SafeMath sub

22, Nov 2018
17.73 ms

Line 54-60 in File SafeMath.sol

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

Line 61-66 in File SafeMath.sol

```
61  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
62    require(b <= a);
63    uint256 c = a - b;
64
65    return c;
66  }
```

The code meets the specification

# Formal Verification Request 56

### SafeMath add

22, Nov 2018
19.14 ms

Line 71-77 in File SafeMath.sol

```
71  /*@CTK "SafeMath add"
72    @post (a + b < a || a + b < b) == __reverted
73    @post !__reverted -> __return == a + b
74    @post !__reverted -> !__has_overflow
75    @post !(__has_buf_overflow)
76    @post !(__has_assertion_failure)
77  */
```

Line 78-83 in File SafeMath.sol

```
78  function add(uint256 a, uint256 b) internal pure returns (uint256) {
79    uint256 c = a + b;
80    require(c >= a);
81
82    return c;
83  }
```

The code meets the specification

# Formal Verification Request 57

**SafeMath div**

22, Nov 2018
16.96 ms

Line 89-95 in File SafeMath.sol

```
89  /*@CTK "SafeMath div"
90    @post b != 0 -> !__reverted
91    @post !__reverted -> __return == a % b
92    @post !__reverted -> !__has_overflow
93    @post !(__has_buf_overflow)
94    @post !(__has_assertion_failure)
95  */
```

Line 96-99 in File SafeMath.sol

```
96  function mod(uint256 a, uint256 b) internal pure returns (uint256) {
97    require(b != 0);
98    return a % b;
99  }
```

The code meets the specification

# Formal Verification Request 58

**If method completes, integer overflow would not happen.**

22, Nov 2018
6.12 ms

Line 25 in File ERC20.sol

```
25  //@CTK NO_OVERFLOW
```

Line 31-33 in File ERC20.sol

```
31  function totalSupply() public view returns (uint256) {
32    return _totalSupply;
33  }
```

The code meets the specification

# Formal Verification Request 59

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

📅 22, Nov 2018
⏱ 0.41 ms

Line 26 in File ERC20.sol

```
26  //@CTK NO_BUF_OVERFLOW
```

Line 31-33 in File ERC20.sol

```
31  function totalSupply() public view returns (uint256) {
32    return _totalSupply;
33  }
```

✅ The code meets the specification

# Formal Verification Request 60

**Method will not encounter an assertion failure.**

📅 22, Nov 2018
⏱ 0.43 ms

Line 27 in File ERC20.sol

```
27  //@CTK NO_ASF
```

Line 31-33 in File ERC20.sol

```
31  function totalSupply() public view returns (uint256) {
32    return _totalSupply;
33  }
```

✅ The code meets the specification

# Formal Verification Request 61

**totalSupply correctness**

📅 22, Nov 2018
⏱ 0.44 ms

Line 28-30 in File ERC20.sol

```
28  /*@CTK "totalSupply correctness"
29    @post __return == _totalSupply
30   */
```

Line 31-33 in File ERC20.sol

```
31  function totalSupply() public view returns (uint256) {
32    return _totalSupply;
33  }
```

✅ The code meets the specification

# Formal Verification Request 62

**If method completes, integer overflow would not happen.**

22, Nov 2018
6.71 ms

Line 40 in File ERC20.sol

```
40  //@CTK NO_OVERFLOW
```

Line 46-48 in File ERC20.sol

```
46  function balanceOf(address owner) public view returns (uint256) {
47    return _balances[owner];
48  }
```

The code meets the specification

# Formal Verification Request 63

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

22, Nov 2018
0.52 ms

Line 41 in File ERC20.sol

```
41  //@CTK NO_BUF_OVERFLOW
```

Line 46-48 in File ERC20.sol

```
46  function balanceOf(address owner) public view returns (uint256) {
47    return _balances[owner];
48  }
```

The code meets the specification

# Formal Verification Request 64

**Method will not encounter an assertion failure.**

22, Nov 2018
0.52 ms

Line 42 in File ERC20.sol

```
42  //@CTK NO_ASF
```

Line 46-48 in File ERC20.sol

```
46  function balanceOf(address owner) public view returns (uint256) {
47    return _balances[owner];
48  }
```

The code meets the specification

# Formal Verification Request 65

**balanceOf correctness**

22, Nov 2018
0.46 ms

Line 43-45 in File ERC20.sol

```
43  /*@CTK "balanceOf correctness"
44    @post __return == _balances[owner]
45   */
```

Line 46-48 in File ERC20.sol

```
46  function balanceOf(address owner) public view returns (uint256) {
47    return _balances[owner];
48  }
```

The code meets the specification

# Formal Verification Request 66

**If method completes, integer overflow would not happen.**

22, Nov 2018
7.48 ms

Line 56 in File ERC20.sol

```
56  //@CTK NO_OVERFLOW
```

Line 62-71 in File ERC20.sol

```
62  function allowance(
63    address owner,
64    address spender
65   )
66    public
67    view
68    returns (uint256)
69  {
70    return _allowed[owner][spender];
71  }
```

The code meets the specification

# Formal Verification Request 67

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

22, Nov 2018
0.4 ms

Line 57 in File ERC20.sol

```
57  //@CTK NO_BUF_OVERFLOW
```

Line 62-71 in File ERC20.sol

```
62  function allowance(
63    address owner,
64    address spender
65   )
66    public
67    view
68    returns (uint256)
69  {
70    return _allowed[owner][spender];
71  }
```

The code meets the specification

# Formal Verification Request 68

**Method will not encounter an assertion failure.**

22, Nov 2018
0.4 ms

Line 58 in File ERC20.sol

```
58  //@CTK NO_ASF
```

Line 62-71 in File ERC20.sol

```
62  function allowance(
63    address owner,
64    address spender
65   )
66    public
67    view
68    returns (uint256)
69  {
70    return _allowed[owner][spender];
71  }
```

The code meets the specification

# Formal Verification Request 69

**allowance correctness**

22, Nov 2018
0.44 ms

Line 59-61 in File ERC20.sol

```
59  /*@CTK "allowance correctness"
60    @post __return == _allowed[owner][spender]
61   */
```

Line 62-71 in File ERC20.sol

```
62  function allowance(
63    address owner,
64    address spender
65   )
66    public
67    view
68    returns (uint256)
69  {
70    return _allowed[owner][spender];
71  }
```

The code meets the specification

# Formal Verification Request 70

**If method completes, integer overflow would not happen.**

22, Nov 2018
219.02 ms

Line 78 in File ERC20.sol

```
78  //@CTK NO_OVERFLOW
```

Line 89-92 in File ERC20.sol

```
89  function transfer(address to, uint256 value) public returns (bool) {
90    _transfer(msg.sender, to, value);
91    return true;
92  }
```

The code meets the specification

# Formal Verification Request 71

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

22, Nov 2018
14.92 ms

Line 79 in File ERC20.sol

```
79  //@CTK NO_BUF_OVERFLOW
```

Line 89-92 in File ERC20.sol

```
89  function transfer(address to, uint256 value) public returns (bool) {
90    _transfer(msg.sender, to, value);
91    return true;
92  }
```

The code meets the specification

# Formal Verification Request 72

**Method will not encounter an assertion failure.**

22, Nov 2018
11.86 ms

Line 80 in File ERC20.sol

```
80  //@CTK NO_ASF
```

Line 89-92 in File ERC20.sol

```
89  function transfer(address to, uint256 value) public returns (bool) {
90    _transfer(msg.sender, to, value);
91    return true;
92  }
```

The code meets the specification

# Formal Verification Request 73

**transfer correctness**

22, Nov 2018
203.71 ms

Line 81-88 in File ERC20.sol

```
81  /*@CTK "transfer correctness"
82    @tag assume_completion
83    @post to != 0x0
84    @post value <= _balances[msg.sender]
85    @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
          value
86    @post to != msg.sender -> __post._balances[to] == _balances[to] + value
87    @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
88   */
```

Line 89-92 in File ERC20.sol

```
89  function transfer(address to, uint256 value) public returns (bool) {
90    _transfer(msg.sender, to, value);
91    return true;
92  }
```

The code meets the specification

# Formal Verification Request 74

**If method completes, integer overflow would not happen.**

22, Nov 2018
18.93 ms

Line 103 in File ERC20.sol

```
103  //@CTK NO_OVERFLOW
```

Line 110-116 in File ERC20.sol

```
110  function approve(address spender, uint256 value) public returns (bool) {
111    require(spender != address(0));
112
113    _allowed[msg.sender][spender] = value;
114    emit Approval(msg.sender, spender, value);
115    return true;
116  }
```

The code meets the specification

# Formal Verification Request 75

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

22, Nov 2018
0.78 ms

Line 104 in File ERC20.sol

```
104  //@CTK NO_BUF_OVERFLOW
```

Line 110-116 in File ERC20.sol

```
110  function approve(address spender, uint256 value) public returns (bool) {
111    require(spender != address(0));
112
113    _allowed[msg.sender][spender] = value;
114    emit Approval(msg.sender, spender, value);
115    return true;
116  }
```

The code meets the specification

# Formal Verification Request 76

**Method will not encounter an assertion failure.**

22, Nov 2018
0.68 ms

Line 105 in File ERC20.sol

```
105  //@CTK NO_ASF
```

Line 110-116 in File ERC20.sol

```
110  function approve(address spender, uint256 value) public returns (bool) {
111    require(spender != address(0));
112
113    _allowed[msg.sender][spender] = value;
114    emit Approval(msg.sender, spender, value);
115    return true;
116  }
```

The code meets the specification

# Formal Verification Request 77

approve correctness

22, Nov 2018
1.86 ms

Line 106-109 in File ERC20.sol

```
106 /*@CTK "approve correctness"
107   @post spender == 0x0 -> __reverted
108   @post spender != 0x0 -> __post._allowed[msg.sender][spender] == value
109 */
```

Line 110-116 in File ERC20.sol

```
110 function approve(address spender, uint256 value) public returns (bool) {
111   require(spender != address(0));
112
113   _allowed[msg.sender][spender] = value;
114   emit Approval(msg.sender, spender, value);
115   return true;
116 }
```

The code meets the specification

# Formal Verification Request 78

If method completes, integer overflow would not happen.

22, Nov 2018
154.12 ms

Line 124 in File ERC20.sol

```
124 //@CTK NO_OVERFLOW
```

Line 136-149 in File ERC20.sol

```
136 function transferFrom(
137   address from,
138   address to,
139   uint256 value
140 )
141   public
142   returns (bool)
143 {
144   require(value <= _allowed[from][msg.sender]);
145
146   _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
147   _transfer(from, to, value);
148   return true;
149 }
```

The code meets the specification

# Formal Verification Request 79

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

22, Nov 2018
8.73 ms

Line 125 in File ERC20.sol

```
125  //@CTK NO_BUF_OVERFLOW
```

Line 136-149 in File ERC20.sol

```
136  function transferFrom(
137    address from,
138    address to,
139    uint256 value
140  )
141    public
142    returns (bool)
143  {
144    require(value <= _allowed[from][msg.sender]);
145
146    _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
147    _transfer(from, to, value);
148    return true;
149  }
```

The code meets the specification

# Formal Verification Request 80

**Method will not encounter an assertion failure.**

22, Nov 2018
8.5 ms

Line 126 in File ERC20.sol

```
126  //@CTK NO_ASF
```

Line 136-149 in File ERC20.sol

```
136  function transferFrom(
137    address from,
138    address to,
139    uint256 value
140  )
141    public
142    returns (bool)
143  {
144    require(value <= _allowed[from][msg.sender]);
145
```

```
146     _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
147     _transfer(from, to, value);
148     return true;
149   }
```

The code meets the specification

# Formal Verification Request 81

**transferFrom correctness**

22, Nov 2018
234.93 ms

Line 127-135 in File ERC20.sol

```
127   /*@CTK "transferFrom correctness"
128     @tag assume_completion
129     @post to != 0x0
130     @post value <= _balances[from] && value <= _allowed[from][msg.sender]
131     @post to != from -> __post._balances[from] == _balances[from] - value
132     @post to != from -> __post._balances[to] == _balances[to] + value
133     @post to == from -> __post._balances[from] == _balances[from]
134     @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
135    */
```

Line 136-149 in File ERC20.sol

```
136   function transferFrom(
137     address from,
138     address to,
139     uint256 value
140   )
141     public
142     returns (bool)
143   {
144     require(value <= _allowed[from][msg.sender]);
145
146     _allowed[from][msg.sender] = _allowed[from][msg.sender].sub(value);
147     _transfer(from, to, value);
148     return true;
149   }
```

The code meets the specification

# Formal Verification Request 82

**If method completes, integer overflow would not happen.**

22, Nov 2018
49.6 ms

Line 160 in File ERC20.sol

```
160   //@CTK NO_OVERFLOW
```

Line 168-181 in File ERC20.sol

```
168  function increaseAllowance(
169    address spender,
170    uint256 addedValue
171  )
172    public
173    returns (bool)
174  {
175    require(spender != address(0));
176
177    _allowed[msg.sender][spender] = (
178      _allowed[msg.sender][spender].add(addedValue));
179    emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
180    return true;
181  }
```

The code meets the specification

# Formal Verification Request 83

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

22, Nov 2018
0.85 ms

Line 161 in File ERC20.sol

```
161  //@CTK NO_BUF_OVERFLOW
```

Line 168-181 in File ERC20.sol

```
168  function increaseAllowance(
169    address spender,
170    uint256 addedValue
171  )
172    public
173    returns (bool)
174  {
175    require(spender != address(0));
176
177    _allowed[msg.sender][spender] = (
178      _allowed[msg.sender][spender].add(addedValue));
179    emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
180    return true;
181  }
```

The code meets the specification

# Formal Verification Request 84

**Method will not encounter an assertion failure.**

22, Nov 2018
0.82 ms

Line 162 in File ERC20.sol

```
162  //@CTK NO_ASF
```

Line 168-181 in File ERC20.sol

```
168  function increaseAllowance(
169    address spender,
170    uint256 addedValue
171  )
172    public
173    returns (bool)
174  {
175    require(spender != address(0));
176
177    _allowed[msg.sender][spender] = (
178      _allowed[msg.sender][spender].add(addedValue));
179    emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
180    return true;
181  }
```

The code meets the specification

# Formal Verification Request 85

**increaseAllowance correctness**

22, Nov 2018
3.19 ms

Line 163-167 in File ERC20.sol

```
163  /*@CTK "increaseAllowance correctness"
164    @tag assume_completion
165    @post spender != 0x0
166    @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
           addedValue
167  */
```

Line 168-181 in File ERC20.sol

```
168  function increaseAllowance(
169    address spender,
170    uint256 addedValue
171  )
172    public
173    returns (bool)
174  {
175    require(spender != address(0));
176
177    _allowed[msg.sender][spender] = (
178      _allowed[msg.sender][spender].add(addedValue));
179    emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
180    return true;
181  }
```

The code meets the specification

# Formal Verification Request 86

**If method completes, integer overflow would not happen.**

📅 22, Nov 2018
⏱ 51.18 ms

Line 192 in File ERC20.sol

```
192  //@CTK NO_OVERFLOW
```

Line 200-213 in File ERC20.sol

```
200  function decreaseAllowance(
201    address spender,
202    uint256 subtractedValue
203  )
204    public
205    returns (bool)
206  {
207    require(spender != address(0));
208
209    _allowed[msg.sender][spender] = (
210      _allowed[msg.sender][spender].sub(subtractedValue));
211    emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
212    return true;
213  }
```

✅ The code meets the specification

# Formal Verification Request 87

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

📅 22, Nov 2018
⏱ 0.81 ms

Line 193 in File ERC20.sol

```
193  //@CTK NO_BUF_OVERFLOW
```

Line 200-213 in File ERC20.sol

```
200  function decreaseAllowance(
201    address spender,
202    uint256 subtractedValue
203  )
204    public
205    returns (bool)
206  {
207    require(spender != address(0));
208
209    _allowed[msg.sender][spender] = (
210      _allowed[msg.sender][spender].sub(subtractedValue));
211    emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
212    return true;
213  }
```

The code meets the specification

# Formal Verification Request 88

**Method will not encounter an assertion failure.**

22, Nov 2018
0.79 ms

Line 194 in File ERC20.sol

```
194   //@CTK NO_ASF
```

Line 200-213 in File ERC20.sol

```
200   function decreaseAllowance(
201     address spender,
202     uint256 subtractedValue
203   )
204     public
205     returns (bool)
206   {
207     require(spender != address(0));
208
209     _allowed[msg.sender][spender] = (
210       _allowed[msg.sender][spender].sub(subtractedValue));
211     emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
212     return true;
213   }
```

The code meets the specification

# Formal Verification Request 89

**decreaseAllowance correctness**

22, Nov 2018
3.06 ms

Line 195-199 in File ERC20.sol

```
195   /*@CTK "decreaseAllowance correctness"
196     @tag assume_completion
197     @post spender != 0x0
198     @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
          subtractedValue
199   */
```

Line 200-213 in File ERC20.sol

```
200   function decreaseAllowance(
201     address spender,
202     uint256 subtractedValue
203   )
204     public
```

```
205     returns (bool)
206   {
207     require(spender != address(0));
208
209     _allowed[msg.sender][spender] = (
210       _allowed[msg.sender][spender].sub(subtractedValue));
211     emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
212     return true;
213   }
```

The code meets the specification

# Formal Verification Request 90

**If method completes, integer overflow would not happen.**

22, Nov 2018
8.81 ms

Line 221 in File ERC20.sol

```
221   //@CTK NO_OVERFLOW
```

Line 232-239 in File ERC20.sol

```
232   function _transfer(address from, address to, uint256 value) internal {
233     require(value <= _balances[from]);
234     require(to != address(0));
235
236     _balances[from] = _balances[from].sub(value);
237     _balances[to] = _balances[to].add(value);
238     emit Transfer(from, to, value);
239   }
```

The code meets the specification

# Formal Verification Request 91

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

22, Nov 2018
7.47 ms

Line 222 in File ERC20.sol

```
222   //@CTK NO_BUF_OVERFLOW
```

Line 232-239 in File ERC20.sol

```
232   function _transfer(address from, address to, uint256 value) internal {
233     require(value <= _balances[from]);
234     require(to != address(0));
235
236     _balances[from] = _balances[from].sub(value);
237     _balances[to] = _balances[to].add(value);
238     emit Transfer(from, to, value);
239   }
```

The code meets the specification

# Formal Verification Request 92

**Method will not encounter an assertion failure.**

22, Nov 2018
7.06 ms

Line 223 in File ERC20.sol

```
223  //@CTK NO_ASF
```

Line 232-239 in File ERC20.sol

```
232  function _transfer(address from, address to, uint256 value) internal {
233    require(value <= _balances[from]);
234    require(to != address(0));
235
236    _balances[from] = _balances[from].sub(value);
237    _balances[to] = _balances[to].add(value);
238    emit Transfer(from, to, value);
239  }
```

The code meets the specification

# Formal Verification Request 93

**_transfer correctness**

22, Nov 2018
181.47 ms

Line 224-231 in File ERC20.sol

```
224  /*@CTK "_transfer correctness"
225    @tag assume_completion
226    @post to != 0x0
227    @post value <= _balances[from]
228    @post to != from -> __post._balances[from] == _balances[from] - value
229    @post to != from -> __post._balances[to] == _balances[to] + value
230    @post to == from -> __post._balances[from] == _balances[from]
231   */
```

Line 232-239 in File ERC20.sol

```
232  function _transfer(address from, address to, uint256 value) internal {
233    require(value <= _balances[from]);
234    require(to != address(0));
235
236    _balances[from] = _balances[from].sub(value);
237    _balances[to] = _balances[to].add(value);
238    emit Transfer(from, to, value);
239  }
```

The code meets the specification

# Formal Verification Request 94

**If method completes, integer overflow would not happen.**

22, Nov 2018
91.57 ms

Line 248 in File ERC20.sol

```
248  //@CTK NO_OVERFLOW
```

Line 257-262 in File ERC20.sol

```
257  function _mint(address account, uint256 value) internal {
258    require(account != 0);
259    _totalSupply = _totalSupply.add(value);
260    _balances[account] = _balances[account].add(value);
261    emit Transfer(address(0), account, value);
262  }
```

The code meets the specification

# Formal Verification Request 95

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

22, Nov 2018
3.91 ms

Line 249 in File ERC20.sol

```
249  //@CTK NO_BUF_OVERFLOW
```

Line 257-262 in File ERC20.sol

```
257  function _mint(address account, uint256 value) internal {
258    require(account != 0);
259    _totalSupply = _totalSupply.add(value);
260    _balances[account] = _balances[account].add(value);
261    emit Transfer(address(0), account, value);
262  }
```

The code meets the specification

# Formal Verification Request 96

**Method will not encounter an assertion failure.**

22, Nov 2018
4.78 ms

Line 250 in File ERC20.sol

```
250  //@CTK NO_ASF
```

Line 257-262 in File ERC20.sol

```
257  function _mint(address account, uint256 value) internal {
258    require(account != 0);
259    _totalSupply = _totalSupply.add(value);
260    _balances[account] = _balances[account].add(value);
261    emit Transfer(address(0), account, value);
262  }
```

The code meets the specification

# Formal Verification Request 97

**_mint correctness**

22, Nov 2018
28.95 ms

Line 251-256 in File ERC20.sol

```
251  /*@CTK "_mint correctness"
252    @tag assume_completion
253    @post account != 0x0
254    @post __post._balances[account] == _balances[account] + value
255    @post __post._totalSupply == _totalSupply + value
256   */
```

Line 257-262 in File ERC20.sol

```
257  function _mint(address account, uint256 value) internal {
258    require(account != 0);
259    _totalSupply = _totalSupply.add(value);
260    _balances[account] = _balances[account].add(value);
261    emit Transfer(address(0), account, value);
262  }
```

The code meets the specification

# Formal Verification Request 98

**If method completes, integer overflow would not happen.**

22, Nov 2018
94.58 ms

Line 270 in File ERC20.sol

```
270  //@CTK NO_OVERFLOW
```

Line 280-287 in File ERC20.sol

```
280  function _burn(address account, uint256 value) internal {
281    require(account != 0);
282    require(value <= _balances[account]);
283
284    _totalSupply = _totalSupply.sub(value);
285    _balances[account] = _balances[account].sub(value);
286    emit Transfer(account, address(0), value);
287  }
```

The code meets the specification

# Formal Verification Request 99

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

22, Nov 2018
7.12 ms

Line 271 in File ERC20.sol

```
271   //@CTK NO_BUF_OVERFLOW
```

Line 280-287 in File ERC20.sol

```
280   function _burn(address account, uint256 value) internal {
281     require(account != 0);
282     require(value <= _balances[account]);
283
284     _totalSupply = _totalSupply.sub(value);
285     _balances[account] = _balances[account].sub(value);
286     emit Transfer(account, address(0), value);
287   }
```

The code meets the specification

# Formal Verification Request 100

**Method will not encounter an assertion failure.**

22, Nov 2018
7.54 ms

Line 272 in File ERC20.sol

```
272   //@CTK NO_ASF
```

Line 280-287 in File ERC20.sol

```
280   function _burn(address account, uint256 value) internal {
281     require(account != 0);
282     require(value <= _balances[account]);
283
284     _totalSupply = _totalSupply.sub(value);
285     _balances[account] = _balances[account].sub(value);
286     emit Transfer(account, address(0), value);
287   }
```

The code meets the specification

# Formal Verification Request 101

_burn correctness

22, Nov 2018
146.16 ms

Line 273-279 in File ERC20.sol

```
273  /*@CTK "_burn correctness"
274    @tag assume_completion
275    @post account != 0x0
276    @post value <= _balances[account]
277    @post __post._balances[account] == _balances[account] - value
278    @post __post._totalSupply == _totalSupply - value
279   */
```

Line 280-287 in File ERC20.sol

```
280  function _burn(address account, uint256 value) internal {
281    require(account != 0);
282    require(value <= _balances[account]);
283
284    _totalSupply = _totalSupply.sub(value);
285    _balances[account] = _balances[account].sub(value);
286    emit Transfer(account, address(0), value);
287  }
```

The code meets the specification

# Formal Verification Request 102

If method completes, integer overflow would not happen.

22, Nov 2018
145.49 ms

Line 296 in File ERC20.sol

```
296  //@CTK NO_OVERFLOW
```

Line 307-315 in File ERC20.sol

```
307  function _burnFrom(address account, uint256 value) internal {
308    require(value <= _allowed[account][msg.sender]);
309
310    // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
       ,
311    // this function needs to emit an event with the updated approval.
312    _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
313      value);
314    _burn(account, value);
315  }
```

The code meets the specification

# Formal Verification Request 103

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

22, Nov 2018
8.15 ms

Line 297 in File ERC20.sol

```
297  //@CTK NO_BUF_OVERFLOW
```

Line 307-315 in File ERC20.sol

```
307  function _burnFrom(address account, uint256 value) internal {
308    require(value <= _allowed[account][msg.sender]);
309
310    // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
         ,
311    // this function needs to emit an event with the updated approval.
312    _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
313      value);
314    _burn(account, value);
315  }
```

The code meets the specification

# Formal Verification Request 104

**Method will not encounter an assertion failure.**

22, Nov 2018
9.08 ms

Line 298 in File ERC20.sol

```
298  //@CTK NO_ASF
```

Line 307-315 in File ERC20.sol

```
307  function _burnFrom(address account, uint256 value) internal {
308    require(value <= _allowed[account][msg.sender]);
309
310    // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
         ,
311    // this function needs to emit an event with the updated approval.
312    _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
313      value);
314    _burn(account, value);
315  }
```

The code meets the specification

# Formal Verification Request 105

**_burnFrom correctness**

22, Nov 2018
269.02 ms

Line 299-306 in File ERC20.sol

```
299  /*@CTK "_burnFrom correctness"
300    @tag assume_completion
301    @post account != 0x0
302    @post value <= _balances[account] && value <= _allowed[account][msg.sender]
303    @post __post._balances[account] == _balances[account] - value
304    @post __post._totalSupply == _totalSupply - value
305    @post __post._allowed[account][msg.sender] == _allowed[account][msg.sender] -
           value
306   */
```

Line 307-315 in File ERC20.sol

```
307  function _burnFrom(address account, uint256 value) internal {
308    require(value <= _allowed[account][msg.sender]);
309
310    // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted
           ,
311    // this function needs to emit an event with the updated approval.
312    _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
313      value);
314    _burn(account, value);
315  }
```

The code meets the specification

# Formal Verification Request 106

**If method completes, integer overflow would not happen.**

22, Nov 2018
11.27 ms

Line 16 in File ERC20Detailed.sol

```
16  //@CTK NO_OVERFLOW
```

Line 24-28 in File ERC20Detailed.sol

```
24  constructor(string name, string symbol, uint8 decimals) public {
25    _name = name;
26    _symbol = symbol;
27    _decimals = decimals;
28  }
```

The code meets the specification

# Formal Verification Request 107

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

📅 22, Nov 2018
⏱ 0.45 ms

Line 17 in File ERC20Detailed.sol

```
17  //@CTK NO_BUF_OVERFLOW
```

Line 24-28 in File ERC20Detailed.sol

```
24  constructor(string name, string symbol, uint8 decimals) public {
25    _name = name;
26    _symbol = symbol;
27    _decimals = decimals;
28  }
```

✅ The code meets the specification

# Formal Verification Request 108

**Method will not encounter an assertion failure.**

📅 22, Nov 2018
⏱ 0.44 ms

Line 18 in File ERC20Detailed.sol

```
18  //@CTK NO_ASF
```

Line 24-28 in File ERC20Detailed.sol

```
24  constructor(string name, string symbol, uint8 decimals) public {
25    _name = name;
26    _symbol = symbol;
27    _decimals = decimals;
28  }
```

✅ The code meets the specification

# Formal Verification Request 109

**ERC20Detailed constructor correctness**

📅 22, Nov 2018
⏱ 0.89 ms

Line 19-23 in File ERC20Detailed.sol

```
19  /*@CTK "ERC20Detailed constructor correctness"
20    @post __post._name == name
21    @post __post._symbol == symbol
22    @post __post._decimals == decimals
23   */
```

Line 24-28 in File ERC20Detailed.sol

```
24  constructor(string name, string symbol, uint8 decimals) public {
25    _name = name;
26    _symbol = symbol;
27    _decimals = decimals;
28  }
```

The code meets the specification

# Formal Verification Request 110

**If method completes, integer overflow would not happen.**

22, Nov 2018
7.73 ms

Line 33 in File ERC20Detailed.sol

```
33  //@CTK NO_OVERFLOW
```

Line 39-41 in File ERC20Detailed.sol

```
39  function name() public view returns(string) {
40    return _name;
41  }
```

The code meets the specification

# Formal Verification Request 111

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

22, Nov 2018
0.4 ms

Line 34 in File ERC20Detailed.sol

```
34  //@CTK NO_BUF_OVERFLOW
```

Line 39-41 in File ERC20Detailed.sol

```
39  function name() public view returns(string) {
40    return _name;
41  }
```

The code meets the specification

# Formal Verification Request 112

**Method will not encounter an assertion failure.**

22, Nov 2018
0.48 ms

Line 35 in File ERC20Detailed.sol

```
35  //@CTK NO_ASF
```

Line 39-41 in File ERC20Detailed.sol

```
39  function name() public view returns(string) {
40    return _name;
41  }
```

The code meets the specification

# Formal Verification Request 113

**ERC20Detailed name correctness**

22, Nov 2018
0.45 ms

Line 36-38 in File ERC20Detailed.sol

```
36  /*@CTK "ERC20Detailed name correctness"
37    @post __return == _name
38   */
```

Line 39-41 in File ERC20Detailed.sol

```
39  function name() public view returns(string) {
40    return _name;
41  }
```

The code meets the specification

# Formal Verification Request 114

**If method completes, integer overflow would not happen.**

22, Nov 2018
8.29 ms

Line 46 in File ERC20Detailed.sol

```
46  //@CTK NO_OVERFLOW
```

Line 52-54 in File ERC20Detailed.sol

```
52  function symbol() public view returns(string) {
53    return _symbol;
54  }
```

The code meets the specification

# Formal Verification Request 115

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

📅 22, Nov 2018
⏱ 0.4 ms

Line 47 in File ERC20Detailed.sol

```
47  //@CTK NO_BUF_OVERFLOW
```

Line 52-54 in File ERC20Detailed.sol

```
52  function symbol() public view returns(string) {
53    return _symbol;
54  }
```

✅ The code meets the specification

# Formal Verification Request 116

**Method will not encounter an assertion failure.**

📅 22, Nov 2018
⏱ 0.4 ms

Line 48 in File ERC20Detailed.sol

```
48  //@CTK NO_ASF
```

Line 52-54 in File ERC20Detailed.sol

```
52  function symbol() public view returns(string) {
53    return _symbol;
54  }
```

✅ The code meets the specification

# Formal Verification Request 117

**ERC20Detailed symbol correctness**

📅 22, Nov 2018
⏱ 0.43 ms

Line 49-51 in File ERC20Detailed.sol

```
49  /*@CTK "ERC20Detailed symbol correctness"
50    @post __return == _symbol
51   */
```

Line 52-54 in File ERC20Detailed.sol

```
52  function symbol() public view returns(string) {
53    return _symbol;
54  }
```

✅ The code meets the specification

# Formal Verification Request 118

**If method completes, integer overflow would not happen.**

📅 22, Nov 2018
⏱ 7.11 ms

Line 56 in File ERC20Detailed.sol

```
56  //@CTK NO_OVERFLOW
```

Line 65-67 in File ERC20Detailed.sol

```
65  function decimals() public view returns(uint8) {
66    return _decimals;
67  }
```

✅ The code meets the specification

# Formal Verification Request 119

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

📅 22, Nov 2018
⏱ 0.53 ms

Line 57 in File ERC20Detailed.sol

```
57  //@CTK NO_BUF_OVERFLOW
```

Line 65-67 in File ERC20Detailed.sol

```
65  function decimals() public view returns(uint8) {
66    return _decimals;
67  }
```

✅ The code meets the specification

# Formal Verification Request 120

**Method will not encounter an assertion failure.**

📅 22, Nov 2018
⏱ 0.63 ms

Line 58 in File ERC20Detailed.sol

```
58  //@CTK NO_ASF
```

Line 65-67 in File ERC20Detailed.sol

```
65  function decimals() public view returns(uint8) {
66    return _decimals;
67  }
```

✅ The code meets the specification

# Formal Verification Request 121

**ERC20Detailed decimals correctness**

📅 22, Nov 2018
⏱ 0.45 ms

Line 59-61 in File ERC20Detailed.sol

```
59  /*@CTK "ERC20Detailed decimals correctness"
60    @post __return == _decimals
61   */
```

Line 65-67 in File ERC20Detailed.sol

```
65  function decimals() public view returns(uint8) {
66    return _decimals;
67  }
```

✅ The code meets the specification

# Formal Verification Request 122

**If method completes, integer overflow would not happen.**

📅 22, Nov 2018
⏱ 361.95 ms

Line 12 in File ERC20Pausable.sol

```
12  //@CTK NO_OVERFLOW
```

Line 24-33 in File ERC20Pausable.sol

```
24  function transfer(
25    address to,
26    uint256 value
27  )
28    public
29    whenNotPaused
30    returns (bool)
31  {
32    return super.transfer(to, value);
33  }
```

✅ The code meets the specification

# Formal Verification Request 123

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

📅 22, Nov 2018
⏱ 10.13 ms

Line 13 in File ERC20Pausable.sol

```
13  //@CTK NO_BUF_OVERFLOW
```

Line 24-33 in File ERC20Pausable.sol

```
24  function transfer(
25    address to,
26    uint256 value
27  )
28    public
29    whenNotPaused
30    returns (bool)
31  {
32    return super.transfer(to, value);
33  }
```

The code meets the specification

# Formal Verification Request 124

**Method will not encounter an assertion failure.**

22, Nov 2018
10.87 ms

Line 14 in File ERC20Pausable.sol

```
14  //@CTK NO_ASF
```

Line 24-33 in File ERC20Pausable.sol

```
24  function transfer(
25    address to,
26    uint256 value
27  )
28    public
29    whenNotPaused
30    returns (bool)
31  {
32    return super.transfer(to, value);
33  }
```

The code meets the specification

# Formal Verification Request 125

**ERC20Pausable transfer correctness**

22, Nov 2018
179.14 ms

Line 15-23 in File ERC20Pausable.sol

```
15  /*@CTK "ERC20Pausable transfer correctness"
16    @tag assume_completion
17    @post to != 0x0
```

```
18    @post value <= _balances[msg.sender]
19    @post _paused == false
20    @post to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
          value
21    @post to != msg.sender -> __post._balances[to] == _balances[to] + value
22    @post to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
23   */
```

Line 24-33 in File ERC20Pausable.sol

```
24  function transfer(
25    address to,
26    uint256 value
27  )
28    public
29    whenNotPaused
30    returns (bool)
31  {
32    return super.transfer(to, value);
33  }
```

The code meets the specification

# Formal Verification Request 126

**If method completes, integer overflow would not happen.**

22, Nov 2018
334.82 ms

Line 35 in File ERC20Pausable.sol

```
35  //@CTK NO_OVERFLOW
```

Line 48-58 in File ERC20Pausable.sol

```
48  function transferFrom(
49    address from,
50    address to,
51    uint256 value
52  )
53    public
54    whenNotPaused
55    returns (bool)
56  {
57    return super.transferFrom(from, to, value);
58  }
```

The code meets the specification

# Formal Verification Request 127

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

22, Nov 2018
10.09 ms

Line 36 in File ERC20Pausable.sol

```
36  //@CTK NO_BUF_OVERFLOW
```

Line 48-58 in File ERC20Pausable.sol

```
48  function transferFrom(
49    address from,
50    address to,
51    uint256 value
52  )
53    public
54    whenNotPaused
55    returns (bool)
56  {
57    return super.transferFrom(from, to, value);
58  }
```

The code meets the specification

# Formal Verification Request 128

**Method will not encounter an assertion failure.**

22, Nov 2018
11.88 ms

Line 37 in File ERC20Pausable.sol

```
37  //@CTK NO_ASF
```

Line 48-58 in File ERC20Pausable.sol

```
48  function transferFrom(
49    address from,
50    address to,
51    uint256 value
52  )
53    public
54    whenNotPaused
55    returns (bool)
56  {
57    return super.transferFrom(from, to, value);
58  }
```

The code meets the specification

# Formal Verification Request 129

**ERC20Pausable transferFrom correctness**

22, Nov 2018
530.86 ms

Line 38-47 in File ERC20Pausable.sol

```
/*@CTK "ERC20Pausable transferFrom correctness"
  @tag assume_completion
  @post to != 0x0
  @post value <= _balances[from] && value <= _allowed[from][msg.sender]
  @post _paused == false
  @post to != from -> __post._balances[from] == _balances[from] - value
  @post to != from -> __post._balances[to] == _balances[to] + value
  @post to == from -> __post._balances[from] == _balances[from]
  @post __post._allowed[from][msg.sender] == _allowed[from][msg.sender] - value
 */
```

Line 48-58 in File ERC20Pausable.sol

```
function transferFrom(
  address from,
  address to,
  uint256 value
)
  public
  whenNotPaused
  returns (bool)
{
  return super.transferFrom(from, to, value);
}
```

The code meets the specification

# Formal Verification Request 130

**If method completes, integer overflow would not happen.**

22, Nov 2018
75.66 ms

Line 60 in File ERC20Pausable.sol

```
//@CTK NO_OVERFLOW
```

Line 69-78 in File ERC20Pausable.sol

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

The code meets the specification

# Formal Verification Request 131

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

22, Nov 2018
0.96 ms

Line 61 in File ERC20Pausable.sol

```
61  //@CTK NO_BUF_OVERFLOW
```

Line 69-78 in File ERC20Pausable.sol

```
69  function approve(
70    address spender,
71    uint256 value
72  )
73    public
74    whenNotPaused
75    returns (bool)
76  {
77    return super.approve(spender, value);
78  }
```

The code meets the specification

# Formal Verification Request 132

**Method will not encounter an assertion failure.**

22, Nov 2018
0.96 ms

Line 62 in File ERC20Pausable.sol

```
62  //@CTK NO_ASF
```

Line 69-78 in File ERC20Pausable.sol

```
69  function approve(
70    address spender,
71    uint256 value
72  )
73    public
74    whenNotPaused
75    returns (bool)
76  {
77    return super.approve(spender, value);
78  }
```

The code meets the specification

# Formal Verification Request 133

**ERC20Pausable approve correctness**

22, Nov 2018
3.91 ms

Line 63-68 in File ERC20Pausable.sol

```
63  /*@CTK "ERC20Pausable approve correctness"
64    @post spender == 0x0 -> __reverted
65    @post _paused -> __reverted
66    @post spender != 0x0 && !_paused
67          -> __post._allowed[msg.sender][spender] == value
68  */
```

Line 69-78 in File ERC20Pausable.sol

```
69  function approve(
70    address spender,
71    uint256 value
72  )
73    public
74    whenNotPaused
75    returns (bool)
76  {
77    return super.approve(spender, value);
78  }
```

The code meets the specification

# Formal Verification Request 134

**If method completes, integer overflow would not happen.**

22, Nov 2018
146.58 ms

Line 80 in File ERC20Pausable.sol

```
80  //@CTK NO_OVERFLOW
```

Line 89-98 in File ERC20Pausable.sol

```
89  function increaseAllowance(
90    address spender,
91    uint addedValue
92  )
93    public
94    whenNotPaused
95    returns (bool success)
96  {
97    return super.increaseAllowance(spender, addedValue);
98  }
```

The code meets the specification

# Formal Verification Request 135

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

22, Nov 2018
1.4 ms

Line 81 in File ERC20Pausable.sol

```
81  //@CTK NO_BUF_OVERFLOW
```

Line 89-98 in File ERC20Pausable.sol

```
89  function increaseAllowance(
90    address spender,
91    uint addedValue
92  )
93    public
94    whenNotPaused
95    returns (bool success)
96  {
97    return super.increaseAllowance(spender, addedValue);
98  }
```

The code meets the specification

# Formal Verification Request 136

**Method will not encounter an assertion failure.**

22, Nov 2018
1.33 ms

Line 82 in File ERC20Pausable.sol

```
82  //@CTK NO_ASF
```

Line 89-98 in File ERC20Pausable.sol

```
89  function increaseAllowance(
90    address spender,
91    uint addedValue
92  )
93    public
94    whenNotPaused
95    returns (bool success)
96  {
97    return super.increaseAllowance(spender, addedValue);
98  }
```

The code meets the specification

# Formal Verification Request 137

**ERC20Pausable increaseAllowance correctness**

22, Nov 2018
45.2 ms

Line 83-88 in File ERC20Pausable.sol

```
83  /*@CTK "ERC20Pausable increaseAllowance correctness"
84    @tag assume_completion
85    @post spender != 0x0
86    @post _paused == false
87    @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] +
          addedValue
88  */
```

Line 89-98 in File ERC20Pausable.sol

```
89  function increaseAllowance(
90    address spender,
91    uint addedValue
92  )
93    public
94    whenNotPaused
95    returns (bool success)
96  {
97    return super.increaseAllowance(spender, addedValue);
98  }
```

The code meets the specification

# Formal Verification Request 138

**If method completes, integer overflow would not happen.**

22, Nov 2018
128.7 ms

Line 100 in File ERC20Pausable.sol

```
100 //@CTK NO_OVERFLOW
```

Line 109-118 in File ERC20Pausable.sol

```
109 function decreaseAllowance(
110   address spender,
111   uint subtractedValue
112 )
113   public
114   whenNotPaused
115   returns (bool success)
116 {
117   return super.decreaseAllowance(spender, subtractedValue);
118 }
```

The code meets the specification

# Formal Verification Request 139

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

22, Nov 2018
1.3 ms

Line 101 in File ERC20Pausable.sol

```
101  //@CTK NO_BUF_OVERFLOW
```

Line 109-118 in File ERC20Pausable.sol

```
109  function decreaseAllowance(
110    address spender,
111    uint subtractedValue
112  )
113    public
114    whenNotPaused
115    returns (bool success)
116  {
117    return super.decreaseAllowance(spender, subtractedValue);
118  }
```

The code meets the specification

# Formal Verification Request 140

**Method will not encounter an assertion failure.**

22, Nov 2018
1.32 ms

Line 102 in File ERC20Pausable.sol

```
102  //@CTK NO_ASF
```

Line 109-118 in File ERC20Pausable.sol

```
109  function decreaseAllowance(
110    address spender,
111    uint subtractedValue
112  )
113    public
114    whenNotPaused
115    returns (bool success)
116  {
117    return super.decreaseAllowance(spender, subtractedValue);
118  }
```

The code meets the specification

# Formal Verification Request 141

**ERC20Pausable decreaseAllowance correctness**

22, Nov 2018
45.24 ms

Line 103-108 in File ERC20Pausable.sol

```
103 /*@CTK "ERC20Pausable decreaseAllowance correctness"
104   @tag assume_completion
105   @post spender != 0x0
106   @post _paused == false
107   @post __post._allowed[msg.sender][spender] == _allowed[msg.sender][spender] -
          subtractedValue
108 */
```

Line 109-118 in File ERC20Pausable.sol

```
109 function decreaseAllowance(
110   address spender,
111   uint subtractedValue
112 )
113   public
114   whenNotPaused
115   returns (bool success)
116 {
117   return super.decreaseAllowance(spender, subtractedValue);
118 }
```

The code meets the specification

# Formal Verification Request 142

**If method completes, integer overflow would not happen.**

22, Nov 2018
16.06 ms

Line 16 in File SafeERC20.sol

```
16 //@CTK NO_OVERFLOW
```

Line 19-27 in File SafeERC20.sol

```
19 function safeTransfer(
20   IERC20 token,
21   address to,
22   uint256 value
23 )
24   internal
25 {
26   require(token.transfer(to, value));
27 }
```

The code meets the specification

# Formal Verification Request 143

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

22, Nov 2018
0.72 ms

Line 17 in File SafeERC20.sol

```
17  //@CTK NO_BUF_OVERFLOW
```

Line 19-27 in File SafeERC20.sol

```
19  function safeTransfer(
20    IERC20 token,
21    address to,
22    uint256 value
23  )
24    internal
25  {
26    require(token.transfer(to, value));
27  }
```

The code meets the specification

# Formal Verification Request 144

**Method will not encounter an assertion failure.**

22, Nov 2018
0.69 ms

Line 18 in File SafeERC20.sol

```
18  //@CTK NO_ASF
```

Line 19-27 in File SafeERC20.sol

```
19  function safeTransfer(
20    IERC20 token,
21    address to,
22    uint256 value
23  )
24    internal
25  {
26    require(token.transfer(to, value));
27  }
```

The code meets the specification

# Formal Verification Request 145

**If method completes, integer overflow would not happen.**

22, Nov 2018
18.42 ms

Line 29 in File SafeERC20.sol

```
29  //@CTK NO_OVERFLOW
```

Line 32-41 in File SafeERC20.sol

```
32  function safeTransferFrom(
33    IERC20 token,
34    address from,
35    address to,
36    uint256 value
37  )
38    internal
39  {
40    require(token.transferFrom(from, to, value));
41  }
```

The code meets the specification

# Formal Verification Request 146

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

22, Nov 2018
0.5 ms

Line 30 in File SafeERC20.sol

```
30  //@CTK NO_BUF_OVERFLOW
```

Line 32-41 in File SafeERC20.sol

```
32  function safeTransferFrom(
33    IERC20 token,
34    address from,
35    address to,
36    uint256 value
37  )
38    internal
39  {
40    require(token.transferFrom(from, to, value));
41  }
```

The code meets the specification

# Formal Verification Request 147

**Method will not encounter an assertion failure.**

22, Nov 2018
0.5 ms

Line 31 in File SafeERC20.sol

```
31  //@CTK NO_ASF
```

Line 32-41 in File SafeERC20.sol

```
32  function safeTransferFrom(
33    IERC20 token,
34    address from,
35    address to,
36    uint256 value
37  )
38    internal
39  {
40    require(token.transferFrom(from, to, value));
41  }
```

The code meets the specification

# Formal Verification Request 148

**If method completes, integer overflow would not happen.**

22, Nov 2018
32.75 ms

Line 43 in File SafeERC20.sol

```
43  //@CTK NO_OVERFLOW
```

Line 46-58 in File SafeERC20.sol

```
46  function safeApprove(
47    IERC20 token,
48    address spender,
49    uint256 value
50  )
51    internal
52  {
53    // safeApprove should only be called when setting an initial allowance,
54    // or when resetting it to zero. To increase and decrease it, use
55    // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
56    require((value == 0) || (token.allowance(msg.sender, spender) == 0));
57    require(token.approve(spender, value));
58  }
```

The code meets the specification

# Formal Verification Request 149

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

22, Nov 2018
1.58 ms

Line 44 in File SafeERC20.sol

```
44  //@CTK NO_BUF_OVERFLOW
```

Line 46-58 in File SafeERC20.sol

```
46  function safeApprove(
47    IERC20 token,
48    address spender,
49    uint256 value
50  )
51    internal
52  {
53    // safeApprove should only be called when setting an initial allowance,
54    // or when resetting it to zero. To increase and decrease it, use
55    // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
56    require((value == 0) || (token.allowance(msg.sender, spender) == 0));
57    require(token.approve(spender, value));
58  }
```

The code meets the specification

# Formal Verification Request 150

**Method will not encounter an assertion failure.**

22, Nov 2018
1.64 ms

Line 45 in File SafeERC20.sol

```
45  //@CTK NO_ASF
```

Line 46-58 in File SafeERC20.sol

```
46  function safeApprove(
47    IERC20 token,
48    address spender,
49    uint256 value
50  )
51    internal
52  {
53    // safeApprove should only be called when setting an initial allowance,
54    // or when resetting it to zero. To increase and decrease it, use
55    // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
56    require((value == 0) || (token.allowance(msg.sender, spender) == 0));
57    require(token.approve(spender, value));
58  }
```

The code meets the specification