CertiK Verification Report For Jobchain

Request Date: 2019-03-11
Revision Date: 2019-03-19
Contract Address: 0x17280DA053596E097604839C61A2eF5efb7d493f

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 Jobchain(the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the "Agreement"). This Report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This Report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiKs prior written consent.

Summary

This audit report summarises the smart contract verification service requested by Jobchain. The goal of this security audit is to guarantee that the audited smart contracts are robust enough to avoid any potential security loopholes.
The result of this report is only a reflection of the source code that was determined in this scope, and of the source code at the time of the audit.

Type of Issues

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

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116

Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	0	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120
tx.origin for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

Vulnerability Details

Critical

No issue found.

Medium

No issue found.

Low

Redundant Code in SafeMath:

- Line `return c;` not needed in `function mul(uint256 _a, uint256 _b)`
- Line `return c;` not needed in `function add(uint256 _a, uint256 _b)`

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 JobToken.sol

```
pragma solidity ^0.4.25;

library SafeMath {

  /**
  * @dev Multiplies two numbers, throws on overflow.
  */
  /*@CTK "SafeMath mul"
    @post (_a > 0) && (((_a * _b) / _a) != _b) -> __reverted
    @post __reverted -> (_a > 0) && (((_a * _b) / _a) != _b)
    @post !__reverted -> c == _a * _b
    @post !__reverted == !__has_overflow
   */
  function mul(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
    // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
    // benefit is lost if 'b' is also tested.
    // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
    if (_a == 0) {
      return 0;
    }

    c = _a * _b;
    require(c / _a == _b);
//    return c;
  }

  /**
  * @dev Integer division of two numbers, truncating the quotient.
  */
  /*@CTK "SafeMath div"
    @post !__reverted -> __return == _a / _b
    @post !__reverted -> !__has_overflow
   */
  function div(uint256 _a, uint256 _b) internal pure returns (uint256) {
    // assert(_b > 0); // Solidity automatically throws when dividing by 0
    // uint256 c = _a / _b;
    // assert(_a == _b * c + _a % _b); // There is no case in which this doesn't hold
    return _a / _b;
  }

  /**
  * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater than
      minuend).
  */
  /*@CTK "SafeMath sub"
    @post (_a < _b) == __reverted
    @post !__reverted -> __return == _a - _b
    @post !__reverted -> !__has_overflow
  */
  function sub(uint256 _a, uint256 _b) internal pure returns (uint256) {
    require(_b <= _a);
    return _a - _b;
  }

```

```
  /**
  * @dev Adds two numbers, throws on overflow.
  */
  /*@CTK "SafeMath add"
    @post (_a + _b < _a || _a + _b < _b) == __reverted
    @post !__reverted -> c == _a + _b
    @post !__reverted -> !__has_overflow
   */
  function add(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
    c = _a + _b;
    require(c >= _a);
//    return c;
  }
}

contract Ownable {
  address public owner;


  event OwnershipRenounced(address indexed previousOwner);
  event OwnershipTransferred(
    address indexed previousOwner,
    address indexed newOwner
  );


  /**
   * @dev The Ownable constructor sets the original `owner` of the contract to the sender
   * account.
   */
  constructor() public {
    owner = msg.sender;
  }

  /**
   * @dev Throws if called by any account other than the owner.
   */
  modifier onlyOwner() {
    require(msg.sender == owner);
    _;
  }

  /**
   * @dev Allows the current owner to relinquish control of the contract.
   * @notice Renouncing to ownership will leave the contract without an owner.
   * It will not be possible to call the functions with the `onlyOwner`
   * modifier anymore.
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_ASF
  /*@CTK renounceOwnership
    @tag assume_completion
    @post owner == msg.sender
    @post __post.owner == address(0)
   */
  function renounceOwnership() public onlyOwner {
    emit OwnershipRenounced(owner);
```

```
    owner = address(0);
  }

  /**
   * @dev Allows the current owner to transfer control of the contract to a newOwner.
   * @param _newOwner The address to transfer ownership to.
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_ASF
  /*@CTK transferOwnership
    @tag assume_completion
    @post owner == msg.sender
   */
  function transferOwnership(address _newOwner) public onlyOwner {
    _transferOwnership(_newOwner);
  }

  /**
   * @dev Transfers control of the contract to a newOwner.
   * @param _newOwner The address to transfer ownership to.
   */
  //@CTK NO_OVERFLOW
  //@CTK NO_ASF
  /*@CTK _transferOwnership
    @tag assume_completion
    @post _newOwner != address(0)
    @post __post.owner == _newOwner
   */
  function _transferOwnership(address _newOwner) internal {
    require(_newOwner != address(0));
    emit OwnershipTransferred(owner, _newOwner);
    owner = _newOwner;
  }
}

contract ERC20Basic {
  function totalSupply() public view returns (uint256);
  function balanceOf(address _who) public view returns (uint256);
  function transfer(address _to, uint256 _value) public returns (bool);
  event Transfer(address indexed from, address indexed to, uint256 value);
}

contract BasicToken is ERC20Basic {
  using SafeMath for uint256;

  mapping(address => uint256) internal balances;

  uint256 internal totalSupply_;

  /**
  * @dev Total number of tokens in existence
  */
  //@CTK NO_OVERFLOW
  //@CTK NO_ASF
  /*@CTK totalSupply
    @post __return == totalSupply_
   */
  function totalSupply() public view returns (uint256) {
```

```
    return totalSupply_;
  }

  /**
  * @dev Transfer token for a specified address
  * @param _to The address to transfer to.
  * @param _value The amount to be transferred.
  */
  //@CTK NO_BUF_OVERFLOW
  //@CTK NO_OVERFLOW
  //@CTK NO_ASF
  /*@CTK transfer_prerequisite
    @post (_to == address(0)) \/ (_value > balances[msg.sender]) -> __reverted == true
  */
  /*@CTK transfer
    @tag assume_completion
    @pre msg.sender != _to
    @post __post.balances[_to] == balances[_to] + _value
    @post __post.balances[msg.sender] == balances[msg.sender] - _value
    @post __return == true
  */
  function transfer(address _to, uint256 _value) public returns (bool) {
    require(_value <= balances[msg.sender]);
    require(_to != address(0));

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

  /**
  * @dev Gets the balance of the specified address.
  * @param _owner The address to query the the balance of.
  * @return An uint256 representing the amount owned by the passed address.
  */
  //@CTK NO_OVERFLOW
  //@CTK NO_ASF
  /*@CTK balanceOf
    @post __return == balances[_owner]
   */
  function balanceOf(address _owner) public view returns (uint256) {
    return balances[_owner];
  }

}

contract ERC20 is ERC20Basic {
  function allowance(address _owner, address _spender)
    public view returns (uint256);

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

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

```
    uint256 value
  );
}

contract StandardToken is ERC20, BasicToken {

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


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

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

  /**
   * @dev Approve the passed address to spend the specified amount of tokens on behalf
       of msg.sender.
   * Beware that changing an allowance with this method brings the risk that someone
       may use both the old
   * and the new allowance by unfortunate transaction ordering. One possible solution
       to mitigate this
   * race condition is to first reduce the spender's allowance to 0 and set the
       desired value afterwards:
```

```
 * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
 * @param _spender The address which will spend the funds.
 * @param _value The amount of tokens to be spent.
 */
//@CTK NO_OVERFLOW
//@CTK NO_ASF
//@CTK NO_BUF_OVERFLOW
/*@CTK approve
  @post __post.allowed[msg.sender][_spender] == _value
  @post __return == true
*/
function approve(address _spender, uint256 _value) public returns (bool) {
  allowed[msg.sender][_spender] = _value;
  emit Approval(msg.sender, _spender, _value);
  return true;
}

/**
 * @dev Function to check the amount of tokens that an owner allowed to a spender.
 * @param _owner address The address which owns the funds.
 * @param _spender address The address which will spend the funds.
 * @return A uint256 specifying the amount of tokens still available for the spender
     .
 */
//@CTK NO_OVERFLOW
//@CTK NO_ASF
//@CTK NO_BUF_OVERFLOW
/*@CTK allowance
  @post __reverted == false
  @post __return == allowed[_owner][_spender]
*/
function allowance(
  address _owner,
  address _spender
 )
  public
  view
  returns (uint256)
{
  return allowed[_owner][_spender];
}

/**
 * @dev Increase the amount of tokens that an owner allowed to a spender.
 * approve should be called when allowed[_spender] == 0. To increment
 * allowed value is better to use this function to avoid 2 calls (and wait until
 * the first transaction is mined)
 * From MonolithDAO Token.sol
 * @param _spender The address which will spend the funds.
 * @param _addedValue The amount of tokens to increase the allowance by.
 */
//@CTK NO_OVERFLOW
//@CTK NO_BUF_OVERFLOW
//@CTK NO_ASF
/*@CTK "increaseApproval correctness"
  @tag assume_completion
  @post __post.allowed[msg.sender][_spender] ==
        allowed[msg.sender][_spender] + _addedValue
```

```
  */
  function increaseApproval(
    address _spender,
    uint256 _addedValue
  )
    public
    returns (bool)
  {
    allowed[msg.sender][_spender] = (
      allowed[msg.sender][_spender].add(_addedValue));
    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    return true;
  }

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

}

contract MintableToken is StandardToken, Ownable {
  event Mint(address indexed to, uint256 amount);
  event MintFinished();

  bool public mintingFinished = false;
```

```


  modifier canMint() {
    require(!mintingFinished);
    _;
  }

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

  /**
   * @dev Function to mint tokens
   * @param _to The address that will receive the minted tokens.
   * @param _amount The amount of tokens to mint.
   * @return A boolean that indicates if the operation was successful.
   */
  /*@CTK mint
    @tag assume_completion
    @post msg.sender == owner
    @post mintingFinished == false
    @post __post.totalSupply_ == totalSupply_ + _amount
    @post __post.balances[_to] == balances[_to] + _amount
   */
  function mint(
    address _to,
    uint256 _amount
  )
    public
    hasMintPermission
    canMint
    returns (bool)
  {
    totalSupply_ = totalSupply_.add(_amount);
    balances[_to] = balances[_to].add(_amount);
    emit Mint(_to, _amount);
    emit Transfer(address(0), _to, _amount);
    return true;
  }

  /**
   * @dev Function to stop minting new tokens.
   * @return True if the operation was successful.
   */
  /*@CTK finishMinting
    @tag assume_completion
    @post mintingFinished == false
    @post __post.mintingFinished == true
   */
  function finishMinting() public onlyOwner canMint returns (bool) {
    mintingFinished = true;
    emit MintFinished();
    return true;
  }
}

contract ReleasableToken is ERC20, Ownable {
```

```
    /* The finalizer contract that allows unlift the transfer limits on this token */
    address public releaseAgent;

    /** A crowdsale contract can release us to the wild if the sale is a success. If false we are are in transfer lock up period.*/
    bool public released = false;

    /** Map of agents that are allowed to transfer tokens regardless of the lock down period. These are crowdsale contracts and possible the team multisig itself. */
    mapping(address => bool) public transferAgents;

    /**
     * Limit token transfer until the crowdsale is over.
     *
     */
    modifier canTransfer(address _sender) {
        require(released || transferAgents[_sender], "For the token to be able to transfer: it's required that the crowdsale is in released state; or the sender is a transfer agent.");
        _;
    }

    /**
     * Set the contract that can call release and make the token transferable.
     *
     * Design choice. Allow reset the release agent to fix fat finger mistakes.
     */
    //@CTK NO_OVERFLOW
    /*@CTK setReleaseAgent
      @tag assume_completion
      @post msg.sender == owner
      @post __post.releaseAgent == addr
     */
    function setReleaseAgent(address addr) public onlyOwner inReleaseState(false) {

        // We don't do interface check here as we might want to a normal wallet address to act as a release agent
        releaseAgent = addr;
    }

    /**
     * Owner can allow a particular address (a crowdsale contract) to transfer tokens despite the lock up period.
     */
    //@CTK NO_OVERFLOW
    /*@CTK setTransferAgent
      @tag assume_completion
      @post msg.sender == owner
      @post __post.transferAgents[addr] == state
     */
    function setTransferAgent(address addr, bool state) public onlyOwner inReleaseState(false) {
        transferAgents[addr] = state;
    }

    /**
```

```
     * One way function to release the tokens to the wild.
     *
     * Can be called only from the release agent that is the final sale contract. It is only called if the crowdsale has been success (first milestone reached).
     */
    //@CTK NO_OVERFLOW
    /*@CTK releaseTokenTransfer
      @tag assume_completion
      @post msg.sender == releaseAgent
      @post __post.released == true
     */
    function releaseTokenTransfer() public onlyReleaseAgent {
        released = true;
    }

    /** The function can be called only before or after the tokens have been released */
    modifier inReleaseState(bool releaseState) {
        require(releaseState == released, "It's required that the state to check aligns with the released flag.");
        _;
    }

    /** The function can be called only by a whitelisted release agent. */
    modifier onlyReleaseAgent() {
        require(msg.sender == releaseAgent, "Message sender is required to be a release agent.");
        _;
    }

    function transfer(address _to, uint _value) public canTransfer(msg.sender) returns (bool success) {
        // Call StandardToken.transfer()
        return super.transfer(_to, _value);
    }

    function transferFrom(address _from, address _to, uint _value) public canTransfer(_from) returns (bool success) {
        // Call StandardToken.transferForm()
        return super.transferFrom(_from, _to, _value);
    }

}

contract UpgradeAgent {

    uint public originalSupply;

    /** Interface marker */
    function isUpgradeAgent() public pure returns (bool) {
        return true;
    }

    function upgradeFrom(address _from, uint256 _value) public;

}

contract UpgradeableToken is StandardToken {
```

```
using SafeMath for uint256;


/** Contract / person who can set the upgrade path. This can be the same as team
    multisig wallet, as what it is with its default value. */
address public upgradeMaster;

/** The next contract where the tokens will be migrated. */
UpgradeAgent public upgradeAgent;

/** How many tokens we have upgraded by now. */
uint256 public totalUpgraded;

/**
 * Upgrade states.
 *
 * - NotAllowed: The child contract has not reached a condition where the upgrade
     can begin
 * - WaitingForAgent: Token allows upgrade, but we don't have a new agent yet
 * - ReadyToUpgrade: The agent is set and the balance holders can upgrade their
     tokens
 *
 */
enum UpgradeState {Unknown, NotAllowed, WaitingForAgent, ReadyToUpgrade}

/**
 * Somebody has upgraded some of his tokens.
 */
event Upgrade(address indexed _from, address indexed _to, uint256 _value);

/**
 * New upgrade agent available.
 */
event UpgradeAgentSet(address agent);

/**
 * Do not allow construction without upgrade master set.
 */
//@CTK NO_OVERFLOW
/*@CTK UpgradeabbleToken
  @post __post.upgradeMaster == _upgradeMaster
 */
constructor(address _upgradeMaster) public {
    upgradeMaster = _upgradeMaster;
}

/**
 * Allow the token holder to upgrade some of their tokens to a new contract.
 */
//CTK NO_OVERFLOW
function upgrade(uint256 value) public {

    UpgradeState state = getUpgradeState();

    require(state == UpgradeState.ReadyToUpgrade, "It's required that the upgrade
        state is ready.");

```

```
        // Validate input value.
        require(value > 0, "The upgrade value is required to be above 0.");

        balances[msg.sender] = balances[msg.sender].sub(value);

        // Take tokens out from circulation
        totalSupply_ = totalSupply_.sub(value);
        totalUpgraded = totalUpgraded.add(value);

        // Upgrade agent reissues the tokens
        upgradeAgent.upgradeFrom(msg.sender, value);
        emit Upgrade(msg.sender, upgradeAgent, value);
    }

    /**
     * Set an upgrade agent that handles
     */
    //CTK NO_OVERFLOW
    function setUpgradeAgent(address agent) external {

        require(canUpgrade(), "It's required to be in canUpgrade() condition when
            setting upgrade agent.");

        require(agent != address(0), "Agent is required to be an non-empty address when
             setting upgrade agent.");

        // Only a master can designate the next agent
        require(msg.sender == upgradeMaster, "Message sender is required to be the
            upgradeMaster when setting upgrade agent.");

        // Upgrade has already begun for an agent
        require(getUpgradeState() != UpgradeState.ReadyToUpgrade, "Upgrade state is
            required to not be upgrading when setting upgrade agent.");

        require(address(upgradeAgent) == address(0), "upgradeAgent once set, cannot be
            reset");

        upgradeAgent = UpgradeAgent(agent);

        // Bad interface
        require(upgradeAgent.isUpgradeAgent(), "The provided updateAgent contract is
            required to be compliant to the UpgradeAgent interface method when setting
            upgrade agent.");

        // Make sure that token supplies match in source and target
        require(upgradeAgent.originalSupply() == totalSupply_, "The provided
            upgradeAgent contract's originalSupply is required to be equivalent to
            existing contract's totalSupply_ when setting upgrade agent.");

        emit UpgradeAgentSet(upgradeAgent);
    }

    /**
     * Get the state of the token upgrade.
     */
    function getUpgradeState() public view returns (UpgradeState) {
        if (!canUpgrade()) return UpgradeState.NotAllowed;
```

```
        else if (address(upgradeAgent) == address(0)) return UpgradeState.
            WaitingForAgent;
        else return UpgradeState.ReadyToUpgrade;
    }

    /**
     * Change the upgrade master.
     *
     * This allows us to set a new owner for the upgrade mechanism.
     */
    //@CTK NO_OVERFLOW
    /*@CTK setUpgradeMaster
      @tag assume_completion
      @post master != address(0)
      @post msg.sender == upgradeMaster
      @post __post.upgradeMaster == master
     */
    function setUpgradeMaster(address master) public {
        require(master != address(0), "The provided upgradeMaster is required to be a
            non-empty address when setting upgrade master.");

        require(msg.sender == upgradeMaster, "Message sender is required to be the
            original upgradeMaster when setting (new) upgrade master.");

        upgradeMaster = master;
    }

    bool canUpgrade_ = true;

    /**
     * Child contract can enable to provide the condition when the upgrade can begin.
     */
    //@CTK NO_OVERFLOW
    /*@CTK canUpgrade
      @post __return == canUpgrade_
     */
    function canUpgrade() public view returns (bool) {
        return canUpgrade_;
    }
}

contract Jobchain is ReleasableToken, MintableToken, UpgradeableToken {

    event UpdatedTokenInformation(string newName, string newSymbol);

    string public name;

    string public symbol;

    uint8 public decimals;

    address public VerificationNodesWallet;
    address public LaunchIncentiveWallet;
    address public capitalReserveWallet;
    address public ecosystemdevelopmentWallet;
    address public InitialFundingWallet;

    /**
```

```
     * Construct the token.
     *
     * This token must be created through a team multisig wallet, so that it is owned by that wallet.
     *
     * @param _name Token name
     * @param _symbol Token symbol - should be all caps
     * @param _initialSupply How many tokens we start with
     * @param _decimals Number of decimal places
     * @param _mintable Are new tokens created over the crowdsale or do we distribute only the initial supply? Note that when the token becomes transferable the minting always ends.
     */
    /*CTK JobToken
      @tag assume_completion
      @pre totalSupply_ > 0
      @post __post.owner == msg.sender
      @post __post.releaseAgent == owner
      @post __post.name == _name
      @post __post.symbol == _symbol
      @post __post.decimals == _decimals
      @post __post.VerificationNodesWallet == _VerificationNodesWallet
      @post __post.LaunchIncentiveWallet == _LaunchIncentiveWallet
      @post __post.capitalReserveWallet == _capitalReserveWallet
      @post __post.ecosystemGrantsReserveWallet == _ecosystemGrantsReserveWallet
      @post __post.InitialFundingWallet == _InitialFundingWallet
      @post !_mintable -> mintingFinished && totalSupply_ > 0
      @post __post.balances[VerificationNodesWallet] == balances[_VerificationNodesWallet] + totalSupply_ * 20 / 100
      @post __post.balances[LaunchIncentiveWallet] == balances[LaunchIncentiveWallet] + totalSupply_ * 25 / 100
      @post __post.balances[capitalReserveWallet] == balances[capitalReserveWallet] + totalSupply_ * 25 / 100
      @post __post.balances[ecosystemGrantsReserveWallet] == balances[ecosystemGrantsReserveWallet] + totalSupply_ * 20 / 100
      @post __post.balances[InitialFundingWallet] == balances[InitialFundingWallet] + totalSupply_ * 10 / 100
     */
    constructor(string _name, string _symbol, uint256 _initialSupply, uint8 _decimals, bool _mintable,
        address _VerificationNodesWallet,
        address _LaunchIncentiveWallet,
        address _capitalReserveWallet,
        address _ecosystemdevelopmentWallet,
        address _InitialFundingWallet)
    public UpgradeableToken(msg.sender) {

        // Create any address, can be transferred
        // to team multisig via changeOwner(),
        // also remember to call setUpgradeMaster()
        owner = msg.sender;
        releaseAgent = owner;

        name = _name;
        symbol = _symbol;

        decimals = _decimals;

```

```
        VerificationNodesWallet = _VerificationNodesWallet;
        LaunchIncentiveWallet = _LaunchIncentiveWallet;
        capitalReserveWallet = _capitalReserveWallet;
        ecosystemdevelopmentWallet = _ecosystemdevelopmentWallet;
        InitialFundingWallet = _InitialFundingWallet;

        if (_initialSupply > 0) {
            require((_initialSupply % 10) == 0, "_initialSupply has to be a mulitple of 10");
            uint256 twentyfivePerCent = _initialSupply.mul(25).div(100);
            uint256 twentyPerCent = _initialSupply.mul(2).div(10);
            uint256 tenPerCent = _initialSupply.div(10);

            mint(VerificationNodesWallet, twentyPerCent);

            mint(LaunchIncentiveWallet, twentyfivePerCent);

            mint(capitalReserveWallet, twentyfivePerCent);

            mint(ecosystemdevelopmentWallet, twentyPerCent);

            mint(InitialFundingWallet, tenPerCent);

        }

        // No more new supply allowed after the token creation
        if (!_mintable) {
            finishMinting();
            require(totalSupply_ > 0, "Total supply is required to be above 0 if the token is not mintable.");
        }

    }

    /**
     * When token is released to be transferable, enforce no new tokens can be created.
     */
    /*CTK releaseTokenTransfer
      @tag assume_completion
      @post __post.mintingFinished == true
     */
    function releaseTokenTransfer() public onlyReleaseAgent {
        mintingFinished = true;
        super.releaseTokenTransfer();
    }

    /**
     * Allow upgrade agent functionality kick in only if the crowdsale was success.
     */
    //@CTK NO_OVERFLOW
    /*@CTK canUpgrade
      @post __return == (released && canUpgrade_)
     */
    function canUpgrade() public view returns (bool) {
        return released && super.canUpgrade();
    }

```

```
    // Total supply
    //@CTK NO_OVERFLOW
    /*@CTK totalSupply
      @tag assume_completion
      @post __return == totalSupply_ - balances[address(0)]
     */
    function totalSupply() public view returns (uint) {
        return totalSupply_.sub(balances[address(0)]);
    }

}
```

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 JobToken.sol

```
1 pragma solidity ^0.4.25;
```

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

Formal Verification Request 1

SafeMath mul

19, Mar 2019
560.86 ms

Line 8-13 in File JobToken.sol

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

Line 14-25 in File JobToken.sol

```
14  function mul(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
15    // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
16    // benefit is lost if 'b' is also tested.
17    // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
18    if (_a == 0) {
19      return 0;
20    }
21
22    c = _a * _b;
23    require(c / _a == _b);
24 //   return c;
25  }
```

The code meets the specification

Formal Verification Request 2

SafeMath div

19, Mar 2019
9.93 ms

Line 30-33 in File JobToken.sol

```
30  /*@CTK "SafeMath div"
31    @post !__reverted -> __return == _a / _b
32    @post !__reverted -> !__has_overflow
33   */
```

Line 34-39 in File JobToken.sol

```
34  function div(uint256 _a, uint256 _b) internal pure returns (uint256) {
35    // assert(_b > 0); // Solidity automatically throws when dividing by 0
36    // uint256 c = _a / _b;
37    // assert(_a == _b * c + _a % _b); // There is no case in which this doesn't hold
38    return _a / _b;
39  }
```

The code meets the specification

Formal Verification Request 3

SafeMath sub

19, Mar 2019
23.6 ms

Line 44-48 in File JobToken.sol

```
44  /*@CTK "SafeMath sub"
45    @post (_a < _b) == __reverted
46    @post !__reverted -> __return == _a - _b
47    @post !__reverted -> !__has_overflow
48  */
```

Line 49-52 in File JobToken.sol

```
49  function sub(uint256 _a, uint256 _b) internal pure returns (uint256) {
50    require(_b <= _a);
51    return _a - _b;
52  }
```

The code meets the specification

Formal Verification Request 4

SafeMath add

19, Mar 2019
31.14 ms

Line 57-61 in File JobToken.sol

```
57  /*@CTK "SafeMath add"
58    @post (_a + _b < _a || _a + _b < _b) == __reverted
59    @post !__reverted -> c == _a + _b
60    @post !__reverted -> !__has_overflow
61   */
```

Line 62-66 in File JobToken.sol

```
62  function add(uint256 _a, uint256 _b) internal pure returns (uint256 c) {
63    c = _a + _b;
64    require(c >= _a);
65 //   return c;
66  }
```

The code meets the specification

Formal Verification Request 5

If method completes, integer overflow would not happen.

19, Mar 2019
20.57 ms

Line 102 in File JobToken.sol

```
102 //@CTK NO_OVERFLOW
```

Line 109-112 in File JobToken.sol

```
109 function renounceOwnership() public onlyOwner {
110   emit OwnershipRenounced(owner);
111   owner = address(0);
112 }
```

✅ The code meets the specification

Formal Verification Request 6

Method will not encounter an assertion failure.

📅 19, Mar 2019
⏱ 0.71 ms

Line 103 in File JobToken.sol

```
103 //@CTK NO_ASF
```

Line 109-112 in File JobToken.sol

```
109 function renounceOwnership() public onlyOwner {
110   emit OwnershipRenounced(owner);
111   owner = address(0);
112 }
```

✅ The code meets the specification

Formal Verification Request 7

renounceOwnership

📅 19, Mar 2019
⏱ 2.11 ms

Line 104-108 in File JobToken.sol

```
104 /*@CTK renounceOwnership
105   @tag assume_completion
106   @post owner == msg.sender
107   @post __post.owner == address(0)
108  */
```

Line 109-112 in File JobToken.sol

```
109 function renounceOwnership() public onlyOwner {
110   emit OwnershipRenounced(owner);
111   owner = address(0);
112 }
```

✅ The code meets the specification

Formal Verification Request 8

If method completes, integer overflow would not happen.

19, Mar 2019
92.0 ms

Line 118 in File JobToken.sol

```
118 //@CTK NO_OVERFLOW
```

Line 124-126 in File JobToken.sol

```
124 function transferOwnership(address _newOwner) public onlyOwner {
125   _transferOwnership(_newOwner);
126 }
```

The code meets the specification

Formal Verification Request 9

Method will not encounter an assertion failure.

19, Mar 2019
0.94 ms

Line 119 in File JobToken.sol

```
119 //@CTK NO_ASF
```

Line 124-126 in File JobToken.sol

```
124 function transferOwnership(address _newOwner) public onlyOwner {
125   _transferOwnership(_newOwner);
126 }
```

The code meets the specification

Formal Verification Request 10

transferOwnership

19, Mar 2019
4.83 ms

Line 120-123 in File JobToken.sol

```
120 /*@CTK transferOwnership
121   @tag assume_completion
122   @post owner == msg.sender
123  */
```

Line 124-126 in File JobToken.sol

```
124 function transferOwnership(address _newOwner) public onlyOwner {
125   _transferOwnership(_newOwner);
126 }
```

The code meets the specification

Formal Verification Request 11

If method completes, integer overflow would not happen.

19, Mar 2019
0.99 ms

Line 132 in File JobToken.sol

```
132 //@CTK NO_OVERFLOW
```

Line 139-143 in File JobToken.sol

```
139 function _transferOwnership(address _newOwner) internal {
140   require(_newOwner != address(0));
141   emit OwnershipTransferred(owner, _newOwner);
142   owner = _newOwner;
143 }
```

The code meets the specification

Formal Verification Request 12

Method will not encounter an assertion failure.

19, Mar 2019
0.88 ms

Line 133 in File JobToken.sol

```
133 //@CTK NO_ASF
```

Line 139-143 in File JobToken.sol

```
139 function _transferOwnership(address _newOwner) internal {
140   require(_newOwner != address(0));
141   emit OwnershipTransferred(owner, _newOwner);
142   owner = _newOwner;
143 }
```

The code meets the specification

Formal Verification Request 13

_transferOwnership

19, Mar 2019
2.28 ms

Line 134-138 in File JobToken.sol

```
134 /*@CTK _transferOwnership
135   @tag assume_completion
136   @post _newOwner != address(0)
137   @post __post.owner == _newOwner
138  */
```

Line 139-143 in File JobToken.sol

```
139 function _transferOwnership(address _newOwner) internal {
140   require(_newOwner != address(0));
141   emit OwnershipTransferred(owner, _newOwner);
142   owner = _newOwner;
143 }
```

The code meets the specification

Formal Verification Request 14

If method completes, integer overflow would not happen.

19, Mar 2019
17.62 ms

Line 163 in File JobToken.sol

```
163 //@CTK NO_OVERFLOW
```

Line 168-170 in File JobToken.sol

```
168 function totalSupply() public view returns (uint256) {
169   return totalSupply_;
170 }
```

The code meets the specification

Formal Verification Request 15

Method will not encounter an assertion failure.

19, Mar 2019
0.6 ms

Line 164 in File JobToken.sol

```
164 //@CTK NO_ASF
```

Line 168-170 in File JobToken.sol

```
168 function totalSupply() public view returns (uint256) {
169   return totalSupply_;
170 }
```

The code meets the specification

Formal Verification Request 16

totalSupply

19, Mar 2019
0.75 ms

Line 165-167 in File JobToken.sol

```
165 /*@CTK totalSupply
166   @post __return == totalSupply_
167  */
```

Line 168-170 in File JobToken.sol

```
168 function totalSupply() public view returns (uint256) {
169   return totalSupply_;
170 }
```

The code meets the specification

Formal Verification Request 17

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

19, Mar 2019
146.53 ms

Line 177 in File JobToken.sol

```
177 //@CTK NO_BUF_OVERFLOW
```

Line 190-198 in File JobToken.sol

```
190 function transfer(address _to, uint256 _value) public returns (bool) {
191   require(_value <= balances[msg.sender]);
192   require(_to != address(0));
193
194   balances[msg.sender] = balances[msg.sender].sub(_value);
195   balances[_to] = balances[_to].add(_value);
196   emit Transfer(msg.sender, _to, _value);
197   return true;
198 }
```

The code meets the specification

Formal Verification Request 18

If method completes, integer overflow would not happen.

19, Mar 2019
16.31 ms

Line 178 in File JobToken.sol

```
178 //@CTK NO_OVERFLOW
```

Line 190-198 in File JobToken.sol

```
190 function transfer(address _to, uint256 _value) public returns (bool) {
191   require(_value <= balances[msg.sender]);
192   require(_to != address(0));
193
194   balances[msg.sender] = balances[msg.sender].sub(_value);
195   balances[_to] = balances[_to].add(_value);
196   emit Transfer(msg.sender, _to, _value);
197   return true;
198 }
```

The code meets the specification

Formal Verification Request 19

Method will not encounter an assertion failure.

19, Mar 2019
12.49 ms

Line 179 in File JobToken.sol

```
179 //@CTK NO_ASF
```

Line 190-198 in File JobToken.sol

```
190 function transfer(address _to, uint256 _value) public returns (bool) {
191   require(_value <= balances[msg.sender]);
192   require(_to != address(0));
193
194   balances[msg.sender] = balances[msg.sender].sub(_value);
195   balances[_to] = balances[_to].add(_value);
196   emit Transfer(msg.sender, _to, _value);
197   return true;
198 }
```

The code meets the specification

Formal Verification Request 20

transfer_prerequisite

19, Mar 2019
28.84 ms

Line 180-182 in File JobToken.sol

```
180 /*@CTK transfer_prerequisite
181   @post (_to == address(0)) \/ (_value > balances[msg.sender]) -> __reverted == true
182 */
```

Line 190-198 in File JobToken.sol

```
190 function transfer(address _to, uint256 _value) public returns (bool) {
191   require(_value <= balances[msg.sender]);
192   require(_to != address(0));
193
194   balances[msg.sender] = balances[msg.sender].sub(_value);
195   balances[_to] = balances[_to].add(_value);
196   emit Transfer(msg.sender, _to, _value);
197   return true;
198 }
```

The code meets the specification

Formal Verification Request 21

transfer

19, Mar 2019
191.3 ms

Line 183-189 in File JobToken.sol

```
183 /*@CTK transfer
184   @tag assume_completion
185   @pre msg.sender != _to
186   @post __post.balances[_to] == balances[_to] + _value
187   @post __post.balances[msg.sender] == balances[msg.sender] - _value
188   @post __return == true
189 */
```

Line 190-198 in File JobToken.sol

```
190 function transfer(address _to, uint256 _value) public returns (bool) {
191   require(_value <= balances[msg.sender]);
192   require(_to != address(0));
193
194   balances[msg.sender] = balances[msg.sender].sub(_value);
195   balances[_to] = balances[_to].add(_value);
196   emit Transfer(msg.sender, _to, _value);
197   return true;
198 }
```

The code meets the specification

Formal Verification Request 22

If method completes, integer overflow would not happen.

19, Mar 2019
11.55 ms

Line 205 in File JobToken.sol

```
205 //@CTK NO_OVERFLOW
```

Line 210-212 in File JobToken.sol

```
210 function balanceOf(address _owner) public view returns (uint256) {
211   return balances[_owner];
212 }
```

The code meets the specification

Formal Verification Request 23

Method will not encounter an assertion failure.

19, Mar 2019
0.94 ms

Line 206 in File JobToken.sol

```
206 //@CTK NO_ASF
```

Line 210-212 in File JobToken.sol

```
210 function balanceOf(address _owner) public view returns (uint256) {
211   return balances[_owner];
212 }
```

The code meets the specification

Formal Verification Request 24

balanceOf

19, Mar 2019
0.8 ms

Line 207-209 in File JobToken.sol

```
207 /*@CTK balanceOf
208   @post __return == balances[_owner]
209  */
```

Line 210-212 in File JobToken.sol

```
210 function balanceOf(address _owner) public view returns (uint256) {
211   return balances[_owner];
212 }
```

The code meets the specification

Formal Verification Request 25

If method completes, integer overflow would not happen.

19, Mar 2019
198.72 ms

Line 242 in File JobToken.sol

```
242 //@CTK NO_OVERFLOW
```

Line 256-273 in File JobToken.sol

```
256 function transferFrom(
257   address _from,
258   address _to,
259   uint256 _value
260 )
261   public
262   returns (bool)
263 {
264   require(_value <= balances[_from]);
265   require(_value <= allowed[_from][msg.sender]);
266   require(_to != address(0));
267
268   balances[_from] = balances[_from].sub(_value);
269   balances[_to] = balances[_to].add(_value);
270   allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
271   emit Transfer(_from, _to, _value);
272   return true;
273 }
```

✅ The code meets the specification

Formal Verification Request 26

Method will not encounter an assertion failure.

📅 19, Mar 2019
⏱ 13.94 ms

Line 243 in File JobToken.sol

```
243 //@CTK NO_ASF
```

Line 256-273 in File JobToken.sol

```
256 function transferFrom(
257   address _from,
258   address _to,
259   uint256 _value
260 )
261   public
262   returns (bool)
263 {
264   require(_value <= balances[_from]);
265   require(_value <= allowed[_from][msg.sender]);
266   require(_to != address(0));
267
268   balances[_from] = balances[_from].sub(_value);
269   balances[_to] = balances[_to].add(_value);
270   allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
271   emit Transfer(_from, _to, _value);
272   return true;
273 }
```

✅ The code meets the specification

Formal Verification Request 27

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

19, Mar 2019
21.58 ms

Line 244 in File JobToken.sol

```
244 //@CTK NO_BUF_OVERFLOW
```

Line 256-273 in File JobToken.sol

```
256 function transferFrom(
257   address _from,
258   address _to,
259   uint256 _value
260 )
261   public
262   returns (bool)
263 {
264   require(_value <= balances[_from]);
265   require(_value <= allowed[_from][msg.sender]);
266   require(_to != address(0));
267
268   balances[_from] = balances[_from].sub(_value);
269   balances[_to] = balances[_to].add(_value);
270   allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
271   emit Transfer(_from, _to, _value);
272   return true;
273 }
```

The code meets the specification

Formal Verification Request 28

transferFrom_prerequisite

19, Mar 2019
13.03 ms

Line 245-247 in File JobToken.sol

```
245 /*@CTK transferFrom_prerequisite
246   @post (_to == address(0)) \/ (_value > balances[_from]) \/ (_value > allowed[_from
        ][msg.sender]) -> __reverted == true
247 */
```

Line 256-273 in File JobToken.sol

```
256 function transferFrom(
257   address _from,
258   address _to,
259   uint256 _value
260 )
261   public
262   returns (bool)
```

```
{
  require(_value <= balances[_from]);
  require(_value <= allowed[_from][msg.sender]);
  require(_to != address(0));

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

✅ The code meets the specification

Formal Verification Request 29

transferFrom

📅 19, Mar 2019
⏱ 517.99 ms

Line 248-255 in File JobToken.sol

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

Line 256-273 in File JobToken.sol

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

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

✅ The code meets the specification

Formal Verification Request 30

If method completes, integer overflow would not happen.

19, Mar 2019
14.6 ms

Line 284 in File JobToken.sol

```
284 //@CTK NO_OVERFLOW
```

Line 291-295 in File JobToken.sol

```
291 function approve(address _spender, uint256 _value) public returns (bool) {
292   allowed[msg.sender][_spender] = _value;
293   emit Approval(msg.sender, _spender, _value);
294   return true;
295 }
```

The code meets the specification

Formal Verification Request 31

Method will not encounter an assertion failure.

19, Mar 2019
0.72 ms

Line 285 in File JobToken.sol

```
285 //@CTK NO_ASF
```

Line 291-295 in File JobToken.sol

```
291 function approve(address _spender, uint256 _value) public returns (bool) {
292   allowed[msg.sender][_spender] = _value;
293   emit Approval(msg.sender, _spender, _value);
294   return true;
295 }
```

The code meets the specification

Formal Verification Request 32

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

19, Mar 2019
0.72 ms

Line 286 in File JobToken.sol

```
286 //@CTK NO_BUF_OVERFLOW
```

Line 291-295 in File JobToken.sol

```
291 function approve(address _spender, uint256 _value) public returns (bool) {
292   allowed[msg.sender][_spender] = _value;
293   emit Approval(msg.sender, _spender, _value);
294   return true;
295 }
```

The code meets the specification

Formal Verification Request 33

approve

19, Mar 2019
2.0 ms

Line 287-290 in File JobToken.sol

```
287 /*@CTK approve
288   @post __post.allowed[msg.sender][_spender] == _value
289   @post __return == true
290 */
```

Line 291-295 in File JobToken.sol

```
291 function approve(address _spender, uint256 _value) public returns (bool) {
292   allowed[msg.sender][_spender] = _value;
293   emit Approval(msg.sender, _spender, _value);
294   return true;
295 }
```

The code meets the specification

Formal Verification Request 34

If method completes, integer overflow would not happen.

19, Mar 2019
11.54 ms

Line 303 in File JobToken.sol

```
303 //@CTK NO_OVERFLOW
```

Line 310-319 in File JobToken.sol

```
310 function allowance(
311   address _owner,
312   address _spender
313  )
314   public
315   view
316   returns (uint256)
317 {
318   return allowed[_owner][_spender];
319 }
```

The code meets the specification

Formal Verification Request 35

Method will not encounter an assertion failure.

19, Mar 2019
0.6 ms

Line 304 in File JobToken.sol

```
304 //@CTK NO_ASF
```

Line 310-319 in File JobToken.sol

```
310 function allowance(
311   address _owner,
312   address _spender
313  )
314   public
315   view
316   returns (uint256)
317 {
318   return allowed[_owner][_spender];
319 }
```

The code meets the specification

Formal Verification Request 36

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

19, Mar 2019
0.83 ms

Line 305 in File JobToken.sol

```
305 //@CTK NO_BUF_OVERFLOW
```

Line 310-319 in File JobToken.sol

```
310 function allowance(
311   address _owner,
312   address _spender
313  )
314   public
315   view
316   returns (uint256)
317 {
318   return allowed[_owner][_spender];
319 }
```

The code meets the specification

Formal Verification Request 37

allowance

19, Mar 2019
0.77 ms

Line 306-309 in File JobToken.sol

```
306 /*@CTK allowance
307   @post __reverted == false
308   @post __return == allowed[_owner][_spender]
309 */
```

Line 310-319 in File JobToken.sol

```
310 function allowance(
311   address _owner,
312   address _spender
313  )
314   public
315   view
316   returns (uint256)
317 {
318   return allowed[_owner][_spender];
319 }
```

The code meets the specification

Formal Verification Request 38

If method completes, integer overflow would not happen.

19, Mar 2019
62.61 ms

Line 330 in File JobToken.sol

```
330 //@CTK NO_OVERFLOW
```

Line 338-349 in File JobToken.sol

```
338 function increaseApproval(
339   address _spender,
340   uint256 _addedValue
341 )
342   public
343   returns (bool)
344 {
345   allowed[msg.sender][_spender] = (
346     allowed[msg.sender][_spender].add(_addedValue));
347   emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
348   return true;
349 }
```

The code meets the specification

Formal Verification Request 39

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

19, Mar 2019
1.0 ms

Line 331 in File JobToken.sol

```
331 //@CTK NO_BUF_OVERFLOW
```

Line 338-349 in File JobToken.sol

```
338 function increaseApproval(
339   address _spender,
340   uint256 _addedValue
341 )
342   public
343   returns (bool)
344 {
345   allowed[msg.sender][_spender] = (
346     allowed[msg.sender][_spender].add(_addedValue));
347   emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
348   return true;
349 }
```

The code meets the specification

Formal Verification Request 40

Method will not encounter an assertion failure.

19, Mar 2019
1.09 ms

Line 332 in File JobToken.sol

```
332 //@CTK NO_ASF
```

Line 338-349 in File JobToken.sol

```
338 function increaseApproval(
339   address _spender,
340   uint256 _addedValue
341 )
342   public
343   returns (bool)
344 {
345   allowed[msg.sender][_spender] = (
346     allowed[msg.sender][_spender].add(_addedValue));
347   emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
348   return true;
349 }
```

The code meets the specification

Formal Verification Request 41

increaseApproval correctness

19, Mar 2019
2.51 ms

Line 333-337 in File JobToken.sol

```
333 /*@CTK "increaseApproval correctness"
334   @tag assume_completion
335   @post __post.allowed[msg.sender][_spender] ==
336         allowed[msg.sender][_spender] + _addedValue
337 */
```

Line 338-349 in File JobToken.sol

```
338 function increaseApproval(
339   address _spender,
340   uint256 _addedValue
341 )
342   public
343   returns (bool)
344 {
345   allowed[msg.sender][_spender] = (
346     allowed[msg.sender][_spender].add(_addedValue));
347   emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
348   return true;
349 }
```

The code meets the specification

Formal Verification Request 42

If method completes, integer overflow would not happen.

19, Mar 2019
69.05 ms

Line 360 in File JobToken.sol

```
360 //@CTK NO_OVERFLOW
```

Line 370-385 in File JobToken.sol

```
370 function decreaseApproval(
371   address _spender,
372   uint256 _subtractedValue
373 )
374   public
375   returns (bool)
376 {
377   uint256 oldValue = allowed[msg.sender][_spender];
378   if (_subtractedValue >= oldValue) {
379     allowed[msg.sender][_spender] = 0;
380   } else {
381     allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
```

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

The code meets the specification

Formal Verification Request 43

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

19, Mar 2019
1.02 ms

Line 361 in File JobToken.sol

```
361 //@CTK NO_BUF_OVERFLOW
```

Line 370-385 in File JobToken.sol

```
370 function decreaseApproval(
371   address _spender,
372   uint256 _subtractedValue
373 )
374   public
375   returns (bool)
376 {
377   uint256 oldValue = allowed[msg.sender][_spender];
378   if (_subtractedValue >= oldValue) {
379     allowed[msg.sender][_spender] = 0;
380   } else {
381     allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
382   }
383   emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
384   return true;
385 }
```

The code meets the specification

Formal Verification Request 44

Method will not encounter an assertion failure.

19, Mar 2019
1.04 ms

Line 362 in File JobToken.sol

```
362 //@CTK NO_ASF
```

Line 370-385 in File JobToken.sol

```
370 function decreaseApproval(
371   address _spender,
372   uint256 _subtractedValue
```

```
373 )
374   public
375   returns (bool)
376 {
377   uint256 oldValue = allowed[msg.sender][_spender];
378   if (_subtractedValue >= oldValue) {
379     allowed[msg.sender][_spender] = 0;
380   } else {
381     allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
382   }
383   emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
384   return true;
385 }
```

The code meets the specification

Formal Verification Request 45

decreaseApproval correctness

19, Mar 2019
13.92 ms

Line 363-369 in File JobToken.sol

```
363 /*@CTK "decreaseApproval correctness"
364   @tag assume_completion
365   @post allowed[msg.sender][_spender] <= _subtractedValue ->
366         __post.allowed[msg.sender][_spender] == 0
367   @post allowed[msg.sender][_spender] > _subtractedValue ->
368         __post.allowed[msg.sender][_spender] == allowed[msg.sender][_spender] -
              _subtractedValue
369 */
```

Line 370-385 in File JobToken.sol

```
370 function decreaseApproval(
371   address _spender,
372   uint256 _subtractedValue
373 )
374   public
375   returns (bool)
376 {
377   uint256 oldValue = allowed[msg.sender][_spender];
378   if (_subtractedValue >= oldValue) {
379     allowed[msg.sender][_spender] = 0;
380   } else {
381     allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
382   }
383   emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
384   return true;
385 }
```

The code meets the specification

Formal Verification Request 46

mint

19, Mar 2019
217.56 ms

Line 412-418 in File JobToken.sol

```
412 /*@CTK mint
413   @tag assume_completion
414   @post msg.sender == owner
415   @post mintingFinished == false
416   @post __post.totalSupply_ == totalSupply_ + _amount
417   @post __post.balances[_to] == balances[_to] + _amount
418  */
```

Line 419-433 in File JobToken.sol

```
419 function mint(
420   address _to,
421   uint256 _amount
422 )
423   public
424   hasMintPermission
425   canMint
426   returns (bool)
427 {
428   totalSupply_ = totalSupply_.add(_amount);
429   balances[_to] = balances[_to].add(_amount);
430   emit Mint(_to, _amount);
431   emit Transfer(address(0), _to, _amount);
432   return true;
433 }
```

The code meets the specification

Formal Verification Request 47

finishMinting

19, Mar 2019
46.88 ms

Line 439-443 in File JobToken.sol

```
439 /*@CTK finishMinting
440   @tag assume_completion
441   @post mintingFinished == false
442   @post __post.mintingFinished == true
443  */
```

Line 444-448 in File JobToken.sol

```
444 function finishMinting() public onlyOwner canMint returns (bool) {
445   mintingFinished = true;
446   emit MintFinished();
```

```
447     return true;
448   }
```

The code meets the specification

Formal Verification Request 48

If method completes, integer overflow would not happen.

19, Mar 2019
53.37 ms

Line 476 in File JobToken.sol

```
476     //@CTK NO_OVERFLOW
```

Line 482-486 in File JobToken.sol

```
482     function setReleaseAgent(address addr) public onlyOwner inReleaseState(false) {
483
484         // We don't do interface check here as we might want to a normal wallet address
                to act as a release agent
485         releaseAgent = addr;
486     }
```

The code meets the specification

Formal Verification Request 49

setReleaseAgent

19, Mar 2019
5.85 ms

Line 477-481 in File JobToken.sol

```
477     /*@CTK setReleaseAgent
478       @tag assume_completion
479       @post msg.sender == owner
480       @post __post.releaseAgent == addr
481      */
```

Line 482-486 in File JobToken.sol

```
482     function setReleaseAgent(address addr) public onlyOwner inReleaseState(false) {
483
484         // We don't do interface check here as we might want to a normal wallet address
                to act as a release agent
485         releaseAgent = addr;
486     }
```

The code meets the specification

Formal Verification Request 50

If method completes, integer overflow would not happen.

19, Mar 2019
61.4 ms

Line 491 in File JobToken.sol

```
491    //@CTK NO_OVERFLOW
```

Line 497-499 in File JobToken.sol

```
497    function setTransferAgent(address addr, bool state) public onlyOwner
           inReleaseState(false) {
498        transferAgents[addr] = state;
499    }
```

The code meets the specification

Formal Verification Request 51

setTransferAgent

19, Mar 2019
8.83 ms

Line 492-496 in File JobToken.sol

```
492    /*@CTK setTransferAgent
493      @tag assume_completion
494      @post msg.sender == owner
495      @post __post.transferAgents[addr] == state
496     */
```

Line 497-499 in File JobToken.sol

```
497    function setTransferAgent(address addr, bool state) public onlyOwner
           inReleaseState(false) {
498        transferAgents[addr] = state;
499    }
```

The code meets the specification

Formal Verification Request 52

If method completes, integer overflow would not happen.

19, Mar 2019
33.97 ms

Line 506 in File JobToken.sol

```
506    //@CTK NO_OVERFLOW
```

Line 512-514 in File JobToken.sol

```
512     function releaseTokenTransfer() public onlyReleaseAgent {
513         released = true;
514     }
```

The code meets the specification

Formal Verification Request 53

releaseTokenTransfer

19, Mar 2019
2.73 ms

Line 507-511 in File JobToken.sol

```
507     /*@CTK releaseTokenTransfer
508       @tag assume_completion
509       @post msg.sender == releaseAgent
510       @post __post.released == true
511      */
```

Line 512-514 in File JobToken.sol

```
512     function releaseTokenTransfer() public onlyReleaseAgent {
513         released = true;
514     }
```

The code meets the specification

Formal Verification Request 54

If method completes, integer overflow would not happen.

19, Mar 2019
10.02 ms

Line 590 in File JobToken.sol

```
590     //@CTK NO_OVERFLOW
```

Line 594-596 in File JobToken.sol

```
594     constructor(address _upgradeMaster) public {
595         upgradeMaster = _upgradeMaster;
596     }
```

The code meets the specification

Formal Verification Request 55

UpgradeabbleToken

19, Mar 2019
0.8 ms

Line 591-593 in File JobToken.sol

```
591 /*@CTK UpgradeabbleToken
592   @post __post.upgradeMaster == _upgradeMaster
593  */
```

Line 594-596 in File JobToken.sol

```
594 constructor(address _upgradeMaster) public {
595     upgradeMaster = _upgradeMaster;
596 }
```

The code meets the specification

Formal Verification Request 56

If method completes, integer overflow would not happen.

19, Mar 2019
75.0 ms

Line 665 in File JobToken.sol

```
665 //@CTK NO_OVERFLOW
```

Line 672-678 in File JobToken.sol

```
672 function setUpgradeMaster(address master) public {
673     require(master != address(0), "The provided upgradeMaster is required to be a
            non-empty address when setting upgrade master.");
674
675     require(msg.sender == upgradeMaster, "Message sender is required to be the
            original upgradeMaster when setting (new) upgrade master.");
676
677     upgradeMaster = master;
678 }
```

The code meets the specification

Formal Verification Request 57

setUpgradeMaster

19, Mar 2019
6.61 ms

Line 666-671 in File JobToken.sol

```
666 /*@CTK setUpgradeMaster
667   @tag assume_completion
668   @post master != address(0)
669   @post msg.sender == upgradeMaster
670   @post __post.upgradeMaster == master
671  */
```

Line 672-678 in File JobToken.sol

```
672    function setUpgradeMaster(address master) public {
673        require(master != address(0), "The provided upgradeMaster is required to be a
               non-empty address when setting upgrade master.");
674
675        require(msg.sender == upgradeMaster, "Message sender is required to be the
               original upgradeMaster when setting (new) upgrade master.");
676
677        upgradeMaster = master;
678    }
```

The code meets the specification

Formal Verification Request 58

If method completes, integer overflow would not happen.

19, Mar 2019
13.57 ms

Line 685 in File JobToken.sol

```
685    //@CTK NO_OVERFLOW
```

Line 689-691 in File JobToken.sol

```
689    function canUpgrade() public view returns (bool) {
690        return canUpgrade_;
691    }
```

The code meets the specification

Formal Verification Request 59

canUpgrade

19, Mar 2019
1.04 ms

Line 686-688 in File JobToken.sol

```
686    /*@CTK canUpgrade
687      @post __return == canUpgrade_
688     */
```

Line 689-691 in File JobToken.sol

```
689    function canUpgrade() public view returns (bool) {
690        return canUpgrade_;
691    }
```

The code meets the specification

Formal Verification Request 60

If method completes, integer overflow would not happen.

19, Mar 2019
59.99 ms

Line 807 in File JobToken.sol

```
807    //@CTK NO_OVERFLOW
```

Line 811-813 in File JobToken.sol

```
811    function canUpgrade() public view returns (bool) {
812        return released && super.canUpgrade();
813    }
```

The code meets the specification

Formal Verification Request 61

canUpgrade

19, Mar 2019
2.94 ms

Line 808-810 in File JobToken.sol

```
808    /*@CTK canUpgrade
809      @post __return == (released && canUpgrade_)
810     */
```

Line 811-813 in File JobToken.sol

```
811    function canUpgrade() public view returns (bool) {
812        return released && super.canUpgrade();
813    }
```

The code meets the specification

Formal Verification Request 62

If method completes, integer overflow would not happen.

19, Mar 2019
59.43 ms

Line 816 in File JobToken.sol

```
816    //@CTK NO_OVERFLOW
```

Line 821-823 in File JobToken.sol

```
821    function totalSupply() public view returns (uint) {
822        return totalSupply_.sub(balances[address(0)]);
823    }
```

The code meets the specification

Formal Verification Request 63

totalSupply

19, Mar 2019
3.03 ms

Line 817-820 in File JobToken.sol

```
817     /*@CTK totalSupply
818       @tag assume_completion
819       @post __return == totalSupply_ - balances[address(0)]
820      */
```

Line 821-823 in File JobToken.sol

```
821     function totalSupply() public view returns (uint) {
822         return totalSupply_.sub(balances[address(0)]);
823     }
```

The code meets the specification