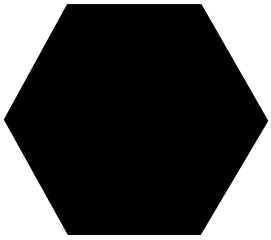
23 NOV 2020

DRC SMART CONTRACT AUDIT REPORT

Hexlant.



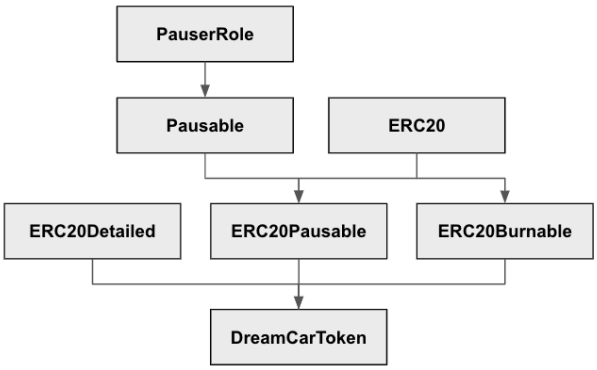
Analysis Purpose

본 리포트는 발행된 컨트랙트 코드가 요구사항을 충분히 만족하는지, 그리고 보안의 취약점과 실제 운영 하면서 발생 할 수 있는 문제들을 파악하고 해결방안을 찾기위해 분석을 수행하고 그 결과를 정리하였습니다. 이번 코드 분석은 다음과 같은 요소들을 검증하기위해 진행하였습니다.

- 구현된 기능의 정상작동 여부
- 기능 수행 중 보안 위험성
- **Off Chain**에서 발생하는 문제에 대한 대비
- 컨트랙트 코드의 가독성 및 코드 완성도

Function Summary

DRC 컨트랙트는 **Open-Zeppelin**에서 제공하는 컨트랙트 코드와 직접 구현한 컨트랙트 코드로 작성되었으며, 다음과 같은 컨트랙트를 통해 **DRC**의 기능을 구현하였습니다.



- **PauserRole**
  컨트랙트 정지관리자 권한에 관련된 기능을 제공합니다. **onlyPauser**을 통해 정지 관리자의 권한 부여 및 제거를 제어할 수 있습니다.

- **Pausable**
  컨트랙트 정지에 관련된 기능을 제공합니다. 컨트랙트가 정지 상태일 경우, 모든 토큰 전송이 이뤄질 수 없도록 상태를 제한할 수 있습니다.

- **ERC20Burnable**
  토큰 소각과 관련된 기능을 제공합니다.

- **DreamCarToken**
  DRC의 메인 컨트랙트입니다. 기본적인 토큰 정보(이름, 심볼, 데시멀, 초기 발행량)을 지정합니다.

Contract

상태 변수와 함수를 포함하여 컨테이너 형태의 계약을 표현하기 위해 사용

Contract	Description
PauserRole	컨트랙트 정지관리자 관련 기능
Pausable	컨트랙트 정지 상태 관련 기능
ERC20	ERC20 표준 인터페이스 관련 기능
ERC20Pausable	컨트랙트 정지에 대한 토큰 전송 관련 기능
ERC20Detailed	토큰 기본 정보 제공 기능
ERC20Burnable	토큰 소각 기능
DreamCarToken	DRC 메인 기능

Interface

컨트랙트 내 구현하고자 하는 표준함수를 정의하기 위해 사용

Interface	Description
IERC20	ERC20 표준 인터페이스

Library

상태 변수를 갖을 수 없고 상속을 지원하지 않는 컨트랙트 라이브러리로, 라이브러리 내 함수가 호출되며 호출한 컨트랙트의 컨텍스트에서 실행

Library	Description
SafeMath	산술연산 제어
Roles	컨트랙트 내 권한 제어

Variable

컨트랙트의 상태를 표현하는 변수들로, 컨트랙트에 필요한 정보들을 저장하기위해 사용

Variable	Description
pausers	정지관리자 주소 해시 테이블
_paused	컨트랙트 정지 상태
_balances	특정 주소의 토큰 잔액 해시 테이블
_allowed	특정 주소에게 출금이 위임된 토큰 잔액 해시 테이블
_totalSupply	토큰 총 발행량
_name	토큰 이름
_symbol	토큰 심볼
_decimals	토큰 최대 표현 가능한 소수점 자리수

Modifier

함수의 한정요소로, 특정 기능을 수행할 때 한정된 조건에서만 실행 될 수 있도록 하기 위해 사용

Modifier	Description
onlyPauser	컨트랙트의 정지 관리자만 실행가능
whenNotPaused	컨트랙트가 정지 상태가 아닐 경우 실행가능
whenPaused	컨트랙트가 정지 상태일 경우 실행가능

Event

컨트랙트 함수 실행에 따른 로그 이벤트로 추후 애플리케이션 적용에 있어 컨트랙트 상황을 보다 쉽게 대응하기 위해 사용

Event	Description
PauserAdded	컨트랙트 정지 관리자 권한 부여 시 이벤트 발생
PauserRemoved	컨트랙트 정지 관리자 권한 제거 시 이벤트 발생
Paused	컨트랙트 정지 시 이벤트 발생
Unpaused	컨트랙트 정지 상태 해제 시 이벤트 발생
Transfer	토큰 전송 시 이벤트 발생
Approval	출금 위임 시 이벤트 발생

Function

컨트랙트의 함수들 로서 컨트랙트에 필요한 특정 로직을 담아 기능 실행을 하기 위해 사용

Function	Description
isPauser	컨트랙트 정지 관리자인지 여부 확인
addPauser	컨트랙트 정지 관리자 권한 부여
renouncePauser	컨트랙트 정지 관리자 권한 포기
_addPauser	컨트랙트 정지 관리자 권한 추가
_removePauser	컨트랙트 정지 관리자 권한 제거
paused	컨트랙트 정지 상태 여부 확인
pause	컨트랙트 정지 상태로 전환
unpause	컨트랙트 비정지 상태로 전환
totalSupply	토큰 총 발행량 확인
balanceOf	특정 주소의 토큰 잔액 확인
allowance	특정 주소에게 출금 위임된 토큰 잔액 확인
transfer	토큰 전송
approve	출금 위임
transferFrom	출금 위임된 토큰 전송
increaseAllowance	출금 위임된 토큰 잔액 증액
decreaseAllowance	출금 위임된 토큰 잔액 감액
_transfer	토큰 전송
_mint	토큰 발행
name	토큰 이름 확인
symbol	토큰 심볼 확인
decimals	토큰 최대 표현 가능한 소수점 자리수 확인
burn	토큰 소각
burnFrom	출금 위임된 토큰 소각

Test Result

Code Coverage

코드 커버리지는 작성한 테스트가 얼마만큼 컨트랙트 코드의 기능들을 테스트 했는지 알 수있는 정량적인 지표입니다.

DRC 컨트랙트는 라이브러리와 일부 컨트랙트에 구현된 기능에 대해 추가적인 호출이 존재하지않습니다.

아래의 **Coverage** 지표는 위 사항을 반영한 결과입니다.

File Name	Statements	Functions	Lines
DRC.sol	100% (94/94)	100% (45/45)	100% (97/97)

Test cases

Test case	Result
토큰 이름은 지정한 이름과 일치한다	PASS
토큰 심볼은 지정한 심볼과 일치한다.	PASS
토큰 데시멀은 지정한 데시멀과 일치한다.	PASS
지정한 초기 발행량이 총 발행량으로 할당된다.	PASS
지정한 초기 발행량이 컨트랙트의 배포 주소에게 할당된다.	PASS
배포 후 배포자 외 주소들의 토큰 잔액은 0이다.	PASS
토큰 전송 시 받는 주소가 0x0 일 경우 예외처리가 되는가?	PASS
토큰 전송 시 보내는 수량이 음수 일 경우 예외처리가 되는가?	PASS
토큰 전송 시 보유한 수량을 초과한 경우 예외처리가 되는가?	PASS
특정 주소에게 출금 위임 시 해당 주소의 출금 위임 잔액이 증액하는가?	PASS
특정 주소에게 출금 위임된 토큰 잔액을 증액 혹은 감액할 수 있는가?	PASS
출금 위임받은 토큰 전송이 가능한가?	PASS
출금 위임받은 토큰 전송 시 관련 주소들의 토큰 잔액이 올바르게 업데이트 되는가?	PASS
출금 위임받은 토큰 전송 시 받는 주소가 0x0일 경우 예외처리가 되는가?	PASS
출금 위임받은 토큰 잔액을 초과한 수량을 전송 시 예외처리가 되는가?	PASS
출금을 위임한 주소의 토큰 보유 잔액이 부족할 경우 예외처리가 되는가?	PASS
컨트랙트 정지관리자 외 주소로부터 컨트랙트 정지 시 예외처리가 되는가?	PASS
컨트랙트가 정지상태일 경우 토큰 전송 시 예외처리가 되는가?	PASS
컨트랙트가 정지상태일 경우 출금 위임된 토큰 전송 시 예외처리가 되는가?	PASS
컨트랙트 정지관리자 외 주소로부터 컨트랙트 정지 해제 시 예외처리가 되는가?	PASS
컨트랙트 정지관리자는 정지 상태를 해제 가능한가?	PASS

Test case	Result
컨트랙트 정지관리자 외 주소로부터 특정 주소에 대해 정지관리자 권한 부여 시 예외처리가 되는가?	PASS
컨트랙트 정지관리자는 특정 주소에게 정지관리자 권한을 부여할 수 있는가?	PASS
컨트랙트 정지관리자는 특정 주소의 정지관리자 권한을 박탈가능한가?	PASS
컨트랙트 정지관리자는 자신의 권한을 포기가능한가?	PASS
보유 잔액이 부족하면 토큰 소각 시 예외처리가 되는가?	PASS
보유 출금 위임 잔액이 부족하면 토큰 소각 시 예외처리가 되는가?	PASS
토큰 소각 시 보유 잔액 및 총 발행량이 함께 감소하는가?	PASS
컨트랙트 배포 시 토큰 전송 이벤트가 발생하는가?	PASS
토큰 전송 시 이벤트가 발생하는가?	PASS
출금 위임 시 이벤트가 발생하는가?	PASS
출금 위임 토큰 잔액 증액 및 감액 시 이벤트가 발생하는가?	PASS
출금 위임된 토큰 전송 시 이벤트가 발생하는가?	PASS
컨트랙트 정지관리자 권한 부여 및 포기 시 이벤트가 발생하는가?	PASS
토큰 소각 시 0x0 주소로의 토큰 전송 이벤트가 발생하는가?	PASS

Vulnerability Analysis

Critical Severity

심각성 치명적 단계는 일반적인 상황에서 보안 또는 큰 문제를 야기 할 수 있는 오류로 반드시 수정해야 하는 항목입니다

해당 항목 없음

High Severity

심각성 높음 단계는 일반적인 상황에서 발생하는 문제는 아니지만, 특수한 조건이나 예외상황에 의해서 문제가 발생 할 수 있는 항목입니다.
추가적인 예외처리나, 코너케이스에 대하여 분석하고 오류를 막을 수 있도록 수정이 필요한 항목입니다.

해당 항목 없음

Medium Severity

심각성 중간단계는 크게 문제가 되는 항목은 아니지만, 좀더 효율적으로 동작 할 수 있도록 수정을 권유하는 항목입니다

해당 항목 없음

Low Severity

낮은 위험도는 성능이나 보안에는 문제는 없지만, 코드 가독성, 컨트랙트 구조 개선을 위해 수정을 권유하는 항목입니다.

해당 항목 없음

Conclusion

DRC 컨트랙트는 검증된 컨트랙트 구현체인 **Openzeppelin**의 컨트랙트를 기반으로 작성되었습니다. **ERC20**의 인터페이스를 모두 충족하였으며 추가적으로 소각 및 정지 기능이 구현되었습니다. 정지기능의 경우, 특정 주소에 한에서 실행 가능하며 실행 시 전체적으로 토큰 전송을 동결할 수 있습니다. 소각 기능의 경우, 누구나 실행 가능하며 토큰의 총 발행량을 감소시킴으로써 토큰의 희소성을 증대시킬 수 있습니다.
컨트랙트의 구현이 단순하며 검증된 코드 기반으로 작성되었기에 보안적 이슈를 발견하지 못하였습니다.

Declare

해당 리포트는 **Hexlant**의 스마트 컨트랙트 보안 감사 결과를 바탕으로 작성되었습니다. 해당 리포트는 비즈니스 모델의 적합성과 법적 규제, 투자에 대한 의견을 보증하지 않습니다. 리포트에 기술한 문제점 이외에 메인넷기술 또는 가상머신을 비롯하여 발견되지 않은 문제점이 있을 수 있습니다. 해당 리포트는 논의 목적으로만 사용됩니다.

```
// File: openzeppelin-solidity/contracts/token/ERC20/IERC20.sol

pragma solidity ^0.4.24;

/**
 * @title ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 */
interface IERC20 {
    function totalSupply() external view returns (uint256);

    function balanceOf(address who) external view returns (uint256);

    function allowance(address owner, address spender)
        external
        view
        returns (uint256);

    function transfer(address to, uint256 value) external returns (bool);

    function approve(address spender, uint256 value) external returns (bool);

    function transferFrom(
        address from,
        address to,
        uint256 value
    ) external returns (bool);

    event Transfer(address indexed from, address indexed to, uint256 value);

    event Approval(
        address indexed owner,
        address indexed spender,
        uint256 value
    );
}

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

pragma solidity ^0.4.24;

/**
 * @title ERC20Detailed token
 * @dev The decimals are only for visualization purposes.
 * All the operations are done using the smallest and indivisible token unit,
 * just as on Ethereum all the operations are done in wei.
 */
contract ERC20Detailed is IERC20 {
    string private _name;
    string private _symbol;
    uint8 private _decimals;

    constructor(
        string name,
        string symbol,
        uint8 decimals
    ) public {
        _name = name;
        _symbol = symbol;
        _decimals = decimals;
    }

    /**
     * @return the name of the token.
     */
    function name() public view returns (string) {
        return _name;
    }

    /**
     * @return the symbol of the token.
     */
    function symbol() public view returns (string) {
        return _symbol;
    }

    /**
     * @return the number of decimals of the token.
     */
    function decimals() public view returns (uint8) {
        return _decimals;
    }
}
```

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

pragma solidity ^0.4.24;

/**
 * @title SafeMath
 * @dev Math operations with safety checks that revert on error
 */
library SafeMath {
    /**
     * @dev Multiplies two numbers, reverts on overflow.
     */
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        // Gas optimization: this is cheaper than requiring '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;
        }

        uint256 c = a * b;
        require(c / a == b);

        return c;
    }

    /**
     * @dev Integer division of two numbers truncating the quotient, reverts on division by zero.
     */
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0); // Solidity only automatically asserts 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 c;
    }

    /**
     * @dev Subtracts two numbers, reverts on overflow (i.e. if subtrahend is greater than minuend).
     */
    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.
     */
    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.
     */
    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;


/**
 * @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
 */
```

```
contract ERC20 is IERC20 {
    using SafeMath for uint256;

    mapping(address => uint256) private _balances;

    mapping(address => mapping(address => uint256)) private _allowed;

    uint256 private _totalSupply;

    /**
     * @dev Total number of tokens in existence
     */
    function totalSupply() public view returns (uint256) {
        return _totalSupply;
    }

    /**
     * @dev Gets the balance of the specified address.
     * @param owner The address to query the balance of.
     * @return An uint256 representing the amount owned by the passed address.
     */
    function balanceOf(address owner) public view returns (uint256) {
        return _balances[owner];
    }

    /**
     * @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.
     */
    function allowance(address owner, address spender)
        public
        view
        returns (uint256)
    {
        return _allowed[owner][spender];
    }

    /**
     * @dev Transfer token for a specified address
     * @param to The address to transfer to.
     * @param value The amount to be transferred.
     */
    function transfer(address to, uint256 value) public returns (bool) {
        _transfer(msg.sender, 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.
     */
    function approve(address spender, uint256 value) public returns (bool) {
        require(spender != address(0));

        _allowed[msg.sender][spender] = value;
        emit Approval(msg.sender, spender, value);
        return true;
    }

    /**
     * @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
     */
  function transferFrom(
        address from,
        address to,
        uint256 value
    ) public returns (bool) {
        require(value <= _allowed[from][msg.sender]);

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

```
/**
   * @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.
   */
  function increaseAllowance(address spender, uint256 addedValue)
      public
      returns (bool)
  {
      require(spender != address(0));

      _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.
   */
  function decreaseAllowance(address spender, uint256 subtractedValue)
      public
      returns (bool)
  {
      require(spender != address(0));

      _allowed[msg.sender][spender] = (
          _allowed[msg.sender][spender].sub(subtractedValue)
      );
      emit Approval(msg.sender, spender, _allowed[msg.sender][spender]);
      return true;
  }

  /**
   * @dev Transfer token for a specified addresses
   * @param from The address to transfer from.
   * @param to The address to transfer to.
   * @param value The amount to be transferred.
   */
  function _transfer(
      address from,
      address to,
      uint256 value
  ) internal {
      require(value <= _balances[from]);
      require(to != address(0));

      _balances[from] = _balances[from].sub(value);
      _balances[to] = _balances[to].add(value);
      emit Transfer(from, to, value);
  }

  /**
   * @dev Internal function that mints an amount of the token and assigns it to
   * an account. This encapsulates the modification of balances such that the
   * proper events are emitted.
   * @param account The account that will receive the created tokens.
   * @param value The amount that will be created.
   */
  function _mint(address account, uint256 value) internal {
      require(account != 0);
      _totalSupply = _totalSupply.add(value);
      _balances[account] = _balances[account].add(value);
      emit Transfer(address(0), account, value);
  }

  /**
   * @dev Internal function that burns an amount of the token of a given
   * account.
   * @param account The account whose tokens will be burnt.
   * @param value The amount that will be burnt.
   */
```

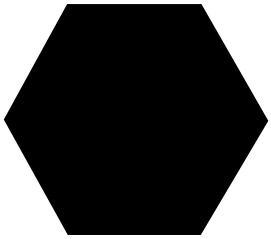
```
   function _burn(address account, uint256 value) internal {
        require(account != 0);
        require(value <= _balances[account]);

        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
    }

    /**
     * @dev Internal function that burns an amount of the token of a given
     * account, deducting from the sender's allowance for said account. Uses the
     * internal burn function.
     * @param account The account whose tokens will be burnt.
     * @param value The amount that will be burnt.
     */
    function _burnFrom(address account, uint256 value) internal {
        require(value <= _allowed[account][msg.sender]);

        // Should https://github.com/OpenZeppelin/zeppelin-solidity/issues/707 be accepted,
        // this function needs to emit an event with the updated approval.
        _allowed[account][msg.sender] = _allowed[account][msg.sender].sub(
            value
        );
        _burn(account, value);
    }
}

// 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
     */
    function add(Role storage role, address account) internal {
        require(account != address(0));
        require(!has(role, account));

        role.bearer[account] = true;
    }

    /**
     * @dev remove an account's access to this role
     */
    function remove(Role storage role, address account) internal {
        require(account != address(0));
        require(has(role, account));

        role.bearer[account] = false;
    }

    /**
     * @dev check if an account has this role
     * @return bool
     */
    function has(Role storage role, address account)
        internal
        view
        returns (bool)
    {
        require(account != address(0));
        return role.bearer[account];
    }
}

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


pragma solidity ^0.4.24;

contract PauserRole {
    using Roles for Roles.Role;

    event PauserAdded(address indexed account);
    event PauserRemoved(address indexed account);
```

```
  Roles.Role private pausers;

    constructor() internal {
        _addPauser(msg.sender);
    }

    modifier onlyPauser() {
        require(isPauser(msg.sender));
        _;
    }

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

    function addPauser(address account) public onlyPauser {
        _addPauser(account);
    }

    function renouncePauser() public {
        _removePauser(msg.sender);
    }

    function _addPauser(address account) internal {
        pausers.add(account);
        emit PauserAdded(account);
    }

    function _removePauser(address account) internal {
        pausers.remove(account);
        emit PauserRemoved(account);
    }
}

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

pragma solidity ^0.4.24;

/**
 * @title Pausable
 * @dev Base contract which allows children to implement an emergency stop mechanism.
 */
contract Pausable is PauserRole {
    event Paused(address account);
    event Unpaused(address account);

    bool private _paused;

    constructor() internal {
        _paused = false;
    }

    /**
     * @return true if the contract is paused, false otherwise.
     */
    function paused() public view returns (bool) {
        return _paused;
    }

    /**
     * @dev Modifier to make a function callable only when the contract is not paused.
     */
    modifier whenNotPaused() {
        require(!_paused);
        _;
    }

    /**
     * @dev Modifier to make a function callable only when the contract is paused.
     */
    modifier whenPaused() {
        require(_paused);
        _;
    }

    /**
     * @dev called by the owner to pause, triggers stopped state
     */
    function pause() public onlyPauser whenNotPaused {
        _paused = true;
        emit Paused(msg.sender);
    }
```

```
    /**
     * @dev called by the owner to unpause, returns to normal state
     */
    function unpause() public onlyPauser whenPaused {
        _paused = false;
        emit Unpaused(msg.sender);
    }
}

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

pragma solidity ^0.4.24;

/**
 * @title Pausable token
 * @dev ERC20 modified with pausable transfers.
 **/
contract ERC20Pausable is ERC20, Pausable {
    function transfer(address to, uint256 value)
        public
        whenNotPaused
        returns (bool)
    {
        return super.transfer(to, value);
    }

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

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

    function increaseAllowance(address spender, uint256 addedValue)
        public
        whenNotPaused
        returns (bool success)
    {
        return super.increaseAllowance(spender, addedValue);
    }

    function decreaseAllowance(address spender, uint256 subtractedValue)
        public
        whenNotPaused
        returns (bool success)
    {
        return super.decreaseAllowance(spender, subtractedValue);
    }
}

// File: openzeppelin-solidity/contracts/token/ERC20/ERC20Burnable.sol

pragma solidity ^0.4.24;

/**
 * @title Burnable Token
 * @dev Token that can be irreversibly burned (destroyed).
 */
contract ERC20Burnable is ERC20 {
    /**
     * @dev Burns a specific amount of tokens.
     * @param value The amount of token to be burned.
     */
    function burn(uint256 value) public {
        _burn(msg.sender, value);
    }

    /**
     * @dev Burns a specific amount of tokens from the target address and decrements allowance
     * @param from address The address which you want to send tokens from
     * @param value uint256 The amount of token to be burned
     */
    function burnFrom(address from, uint256 value) public {
        _burnFrom(from, value);
    }
}
```

```
// File: contracts/token/DreamCarToken.sol

pragma solidity ^0.4.24;

contract DreamCarToken is ERC20Detailed, ERC20Pausable, ERC20Burnable {
    /**
     * @dev constructor to mint initial tokens
     * @param name string
     * @param symbol string
     * @param decimals uint8
     * @param initialSupply uint256
     */
    constructor(
        string name,
        string symbol,
        uint8 decimals,
        uint256 initialSupply
    ) public ERC20Detailed(name, symbol, decimals) {
        // Mint the initial supply
        require(initialSupply > 0, "initialSupply must be greater than zero.");
        _mint(msg.sender, initialSupply * (10**uint256(decimals)));
    }
}
```

Hexlant.
Blockchain Lab
-
contact@hexlant.com
www.hexlant.com



HEXLANT CONTRACT CERTIFICATION

This contract speicifics that it has been validated by the Hexlant Technical Team and notifies that it has not any technical defects.

PUBLISHIED INFORMATION

REPORT NUMBER	ERC20201123
DATE	2020/11/23
PUBLISHIER	SEONGEUN CHO eun@hexlant.com

TOKEN INFORMATION

TOKEN NAME	Dream Car		
SYMBOL	DRC		
PLATFORM	ETHEREUM	**TOKEN TYPE**	ERC-20
TOTAL SUPPLY	1,000,000,000 DRC		
CONTRACT ADDRESS	0xd7f5cabdf696d7d1bf384d7688926a4bdb092c67		

VULNERABLILLITY ANALYSIS

CRITICAL	0	No relevant provision
HIGH	0	No relevant provision
MEDIUM	0	No relevant provision
LOW	0	No relevant provision

CENTRALIZED FUNCTIONS

FREEZE	NO	Ability to freeze tokens in accounts. (The administrator can freeze the hacker's account in case of hacking.)
PAUSE	YES	Ability to pause functions related to token transmission in a contract. (This is used when the administrator needs to prevent the movement of assets due to token swaps or hacking.)
LOCKUP	NO	Ability to block token transfers for a period of time (Administrators can use to set lockout periods for investors, team members, advisors, etc.)
BURN	YES	Ability to reduce total supply by burning tokens
MINT	NO	Ability to increase total supply by minting tokens

Certified by Hexlant._____