智能合约安全审计报告

慢雾安全团队于 2018-07-11 日，收到 NRC 团队对 R 项目智能合约安全审计申请。如下为本次智能合约安全审计细节及结果：

Token 名称：

R

合约地址：

0x7D8b9F24320Dab5369144Eb46927667f4a58dC49

链接地址：

https://etherscan.io/address/0x7D8b9F24320Dab5369144Eb46927667f4a58dC49#code

本次审计项及结果：

（其他未知安全漏洞不包含在本次审计责任范围）

序号	审计大类	审计子类	审计结果
1	溢出审计	-	通过
2	条件竞争审计	-	通过
3	权限控制审计	权限漏洞审计	通过
		权限过大审计	通过
4	安全设计审计	Zeppelin 模块使用安全	通过
		编译器版本安全	通过
		硬编码地址安全	通过
		Fallback 函数使用安全	通过
		显现编码安全	通过
		函数返回值安全	**不通过**
		call 调用安全	通过
5	拒绝服务审计	-	通过
6	Gas 优化审计	-	通过
7	设计逻辑审计	-	通过
8	“假充值”漏洞审计	-	通过

9	恶意 Event 事件日志审计	-	通过

备注：审计意见及建议见代码注释 **//SlowMist//.....**

审计结果：**通过(良)**

审计编号：0X001807130002

审计日期：2018 年 07 月 13 日

审计团队：慢雾安全团队

总结：此为代币(token)合约，同时包含锁仓(tokenVault)部分。综合评估：1、合约在对接钱包、中心化交易所时无风险，当对接去中心化 DApp 时，可能存在兼容性问题，具体为：外部合约使用的 solidity 编译器<0.4.22 版本，跨合约调用不会出现异常；但当外部合约使用的 solidity 编译器>=0.4.22 版本后，调用 transfer 函数将发生 revert；2、合约缺失 totalSupply、transferFrom、approve、allowance 函数，请严格按照 EIP20 规范补充实现。

合约源代码如下：

```
pragma solidity ^ 0.4.18;


//SlowMist// 合约不存在溢出、条件竞争问题

//SlowMist// 使用了大量 OpenZeppelin 的 SafeMath 及 ERC20 标准模块，值得称赞的做法

/**
 * @title Owned
 * @dev The Owned contract has an owner address, and provides basic authorization control
 */
contract Owned {
    address public owner;

    /*Set owner of the contract*/
    function Owned() public {
```

```
        owner = msg.sender;
    }

    /*only owner can be modifier*/
    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }
}

/**
 * @title Pausable
 * @dev Base contract which allows children to implement an emergency stop mechanism.
 */
contract Pausable is Owned {
  event Pause();
  event Unpause();

  bool public paused = false;


  /**
   * @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 onlyOwner whenNotPaused {
    paused = true;
```

```
    Pause();
  }

  /**
   * @dev called by the owner to unpause, returns to normal state
   */
  function unpause() public onlyOwner whenPaused {
    paused = false;
    Unpause();
  }
}

/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
library SafeMath {
    /**
     * @dev Multiplies two numbers, throws on overflow.
     */
    function mul(uint256 a, uint256 b) internal pure returns(uint256) {
        if (a == 0) {
            return 0;
        }
        uint256 c = a * b;
        assert(c / a == b);
        return c;
    }

    /**
     * @dev Integer division of two numbers, truncating the quotient.
     */
    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 c;
    }

    /**
     * @dev Substracts two numbers, throws on overflow (i.e. if subtrahend is greater than minuend).
     */
```

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

    /**
     * @dev Adds two numbers, throws on overflow.
     */
    function add(uint256 a, uint256 b) internal pure returns(uint256) {
        uint256 c = a + b;
        assert(c >= a);
        return c;
    }

}

/*ERC20*/
contract TokenERC20 is Pausable {
    using SafeMath for uint256;
    // Public variables of the token
    string public name = "NRC";
    string public symbol = "R";
    uint8 public decimals = 0;
    // how many token units a buyer gets per wei
    uint256 public rate = 50000;
    // address where funds are collected
    address public wallet = 0xd3C8326064044c36B73043b009155a59e92477D0;
    // contributors address
    address public contributorsAddress = 0xa7db53CB73DBe640DbD480a928dD06f03E2aE7Bd;
    // company address
    address public companyAddress = 0x9c949b51f2CafC3A5efc427621295489B63D861D;
    // market Address
    address public marketAddress = 0x199EcdFaC25567eb4D21C995B817230050d458d9;
    // share of all token
    uint8 public constant ICO_SHARE = 20;
    uint8 public constant CONTRIBUTORS_SHARE = 30;
    uint8 public constant COMPANY_SHARE = 20;
    uint8 public constant MARKET_SHARE = 30;
    // unfronzen periods
    uint8 constant COMPANY_PERIODS = 10;
    uint8 constant CONTRIBUTORS_PERIODS = 3;
    // token totalsupply amount
```

//SlowMist// 这个变量建议改名为 totalSupply，或者新增 totalSupply 函数，返回 Token 总量，EIP20 规范：https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md#totalsupply

```
    uint256 public constant TOTAL_SUPPLY = 80000000000;
    // ico token amount
    uint256 public icoTotalAmount = 16000000000;
    uint256 public companyPeriodsElapsed;
    uint256 public contributorsPeriodsElapsed;
    // token frozened amount
    uint256 public frozenSupply;
    uint256 public initDate;
    uint8 public contributorsCurrentPeriod;
    uint8 public companyCurrentPeriod;
    // This creates an array with all balances
    mapping(address => uint256) public balanceOf;

    // This generates a public event on the blockchain that will notify clients
    event Transfer(address indexed from, address indexed to, uint256 value);
    event InitialToken(string desc, address indexed target, uint256 value);

    /**
     * Constrctor function
     * Initializes contract with initial supply tokens to the creator of the contract
     */
    function TokenERC20(
    ) public {
        // contributors share 30% of totalSupply,but get all by 3 years
        uint256 tempContributors =
TOTAL_SUPPLY.mul(CONTRIBUTORS_SHARE).div(100).div(CONTRIBUTORS_PERIODS);
        contributorsPeriodsElapsed = tempContributors;
        balanceOf[contributorsAddress] = tempContributors;
        InitialToken("contributors", contributorsAddress, tempContributors);

        // company shares 20% of totalSupply,but get all by 10 years
        uint256 tempCompany = TOTAL_SUPPLY.mul(COMPANY_SHARE).div(100).div(COMPANY_PERIODS);
        companyPeriodsElapsed = tempCompany;
        balanceOf[companyAddress] = tempCompany;
        InitialToken("company", companyAddress, tempCompany);

        // ico takes 20% of totalSupply
        uint256 tempIco = TOTAL_SUPPLY.mul(ICO_SHARE).div(100);
```

```
        icoTotalAmount = tempIco;

        // expand the market cost 30% of totalSupply
        uint256 tempMarket = TOTAL_SUPPLY.mul(MARKET_SHARE).div(100);
        balanceOf[marketAddress] = tempMarket;
        InitialToken("market", marketAddress, tempMarket);

        // frozenSupply waitting for being unfrozen
        uint256 tempFrozenSupply =
TOTAL_SUPPLY.sub(tempContributors).sub(tempIco).sub(tempCompany).sub(tempMarket);
        frozenSupply = tempFrozenSupply;
        initDate = block.timestamp;
        contributorsCurrentPeriod = 1;
        companyCurrentPeriod = 1;
        paused = true;
    }
```

//SlowMist// 后面的 NRCToken 合约中完全重写了_transfer 函数，此处可省略

```
    /**
     * Internal transfer, only can be called by this contract
     */
    function _transfer(address _from, address _to, uint _value) internal {
        // Prevent transfer to 0x0 address. Use burn() instead
        require(_to != 0x0);
        // Check if the sender has enough
        require(balanceOf[_from] >= _value);
        // Check for overflows
        require(balanceOf[_to].add(_value) > balanceOf[_to]);
        // Save this for an assertion in the future
        uint previousBalances = balanceOf[_from].add(balanceOf[_to]);
        // Subtract from the sender
        balanceOf[_from] = balanceOf[_from].sub(_value);
        // Add the same to the recipient
        balanceOf[_to] = balanceOf[_to].add(_value);
        Transfer(_from, _to, _value);
        // Asserts are used to use static analysis to find bugs in your code. They should never fail
        assert(balanceOf[_from].add(balanceOf[_to]) == previousBalances);
    }
```

//SlowMist// 后面的 NRCToken 合约中完全重写了 transfer 函数，此处可省略

```
    /**
     * Transfer tokens
     *
     * Send `_value` tokens to `_to` from your account
     *
     * @param _to The address of the recipient
     * @param _value the amount to send
     */
    function transfer(address _to, uint256 _value) public {
        _transfer(msg.sender, _to, _value);
    }
}
/******************************************/
/*       NRCToken STARTS HERE       */
/******************************************/

contract NRCToken is Owned, TokenERC20 {
    uint256 private etherChangeRate = 10 ** 18;
    uint256 private minutesOneYear = 365*24*60 minutes;
    bool public  tokenSaleActive = true;
    // token have been sold
    uint256 public totalSoldToken;
    // all frozenAccount addresses
    mapping(address => bool) public frozenAccount;

    /* This generates a public log event on the blockchain that will notify clients */
    event LogFrozenAccount(address target, bool frozen);
    event LogUnfrozenTokens(string desc, address indexed targetaddress, uint256 unfrozenTokensAmount);
    event LogSetTokenPrice(uint256 tokenPrice);
    event TimePassBy(string desc, uint256 times );
    /**
     * event for token purchase logging
     * @param purchaser who paid for the tokens
     * @param value ehter paid for purchase
     * @param amount amount of tokens purchased
     */
    event LogTokenPurchase(address indexed purchaser, uint256 value, uint256 amount);
    // ICO finished Event
    event TokenSaleFinished(string desc, address indexed contributors, uint256 icoTotalAmount, uint256
totalSoldToken, uint256 leftAmount);

    /* Initializes contract with initial supply tokens to the creator of the contract */
```

```
    function NRCToken() TokenERC20() public {}

    /* Internal transfer, only can be called by this contract */
    function _transfer(address _from, address _to, uint _value) internal {
        require(_from != _to);
        require(_to != 0x0); // Prevent transfer to 0x0 address. Use burn() instead
        require(balanceOf[_from] >= _value); // Check if the sender has enough
        require(balanceOf[_to].add(_value) > balanceOf[_to]); // Check for overflows
        require(!frozenAccount[_from]); // Check if sender is frozen
        require(!frozenAccount[_to]); // Check if recipient is frozen
        balanceOf[_from] = balanceOf[_from].sub(_value); // Subtract from the sender
        balanceOf[_to] = balanceOf[_to].add(_value); // Add the same to the recipient
        Transfer(_from, _to, _value);
    }
       /**
     * Transfer tokens
     *
     * Send `_value` tokens to `_to` from your account
     *
     * @param _to The address of the recipient
     * @param _value the amount to send
     */
    function transfer(address _to, uint256 _value) public {
        _transfer(msg.sender, _to, _value);
```

//SlowMist// 缺少布尔返回值，不符合 EIP20 规范

```
    }
```

//SlowMist// 缺失 transferFrom、approve、allowance 函数，请严格按照 EIP20 规范补充实现：https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md#transferfrom

```
    /// @notice `freeze? Prevent | Allow` `target` from sending & receiving tokens
    /// @param target Address to be frozen
    /// @param freeze either to freeze it or not
    function freezeAccount(address target, bool freeze) public onlyOwner whenNotPaused {
        require(target != 0x0);
        require(target != owner);
        require(frozenAccount[target] != freeze);
        frozenAccount[target] = freeze;
        LogFrozenAccount(target, freeze);
```

```
}

/// @notice Allow users to buy tokens for `newTokenRate` eth
/// @param newTokenRate Price users can buy from the contract
function setPrices(uint256 newTokenRate) public onlyOwner whenNotPaused {
    require(newTokenRate > 0);
    require(newTokenRate <= icoTotalAmount);
    require(tokenSaleActive);
    rate = newTokenRate;
    LogSetTokenPrice(newTokenRate);
}

/// @notice Buy tokens from contract by sending ether
function buy() public payable whenNotPaused {
    // if ICO finished ,can not buy any more!
    require(!frozenAccount[msg.sender]);
    require(tokenSaleActive);
    require(validPurchase());
    uint tokens = getTokenAmount(msg.value); // calculates the amount
    require(!validSoldOut(tokens));
    LogTokenPurchase(msg.sender, msg.value, tokens);
    balanceOf[msg.sender] = balanceOf[msg.sender].add(tokens);
    calcTotalSoldToken(tokens);
    forwardFunds();
}

// Override this method to have a way to add business logic to your crowdsale when buying
function getTokenAmount(uint256 etherAmount) internal view returns(uint256) {
    uint256 temp = etherAmount.mul(rate);
    uint256 amount = temp.div(etherChangeRate);
    return amount;
}

// send ether to the funder wallet
function forwardFunds() internal {
    wallet.transfer(msg.value);
}

// calc totalSoldToken
function calcTotalSoldToken(uint256 soldAmount) internal {
    totalSoldToken = totalSoldToken.add(soldAmount);
    if (totalSoldToken >= icoTotalAmount) {
```

```
            tokenSaleActive = false;
        }
    }

    // @return true if the transaction can buy tokens
    function validPurchase() internal view returns(bool) {
        bool limitPurchase = msg.value >= 1 ether;
        bool isNotTheOwner = msg.sender != owner;
        bool isNotTheCompany = msg.sender != companyAddress;
        bool isNotWallet = msg.sender != wallet;
        bool isNotContributors = msg.sender != contributorsAddress;
        bool isNotMarket = msg.sender != marketAddress;
        return limitPurchase && isNotTheOwner && isNotTheCompany && isNotWallet && isNotContributors &&
isNotMarket;
    }

    // @return true if the ICO is in progress.
    function validSoldOut(uint256 soldAmount) internal view returns(bool) {
        return totalSoldToken.add(soldAmount) > icoTotalAmount;
    }
    // @return current timestamp
    function time() internal constant returns (uint) {
        return block.timestamp;
    }

    /// @dev send the rest of the tokens after the crowdsale end and
    /// send to contributors address
    function finaliseICO() public onlyOwner whenNotPaused {
        require(tokenSaleActive == true);
        uint256 tokensLeft = icoTotalAmount.sub(totalSoldToken);
        tokenSaleActive = false;
        require(tokensLeft > 0);
        balanceOf[contributorsAddress] = balanceOf[contributorsAddress].add(tokensLeft);
        TokenSaleFinished("finaliseICO",     contributorsAddress,     icoTotalAmount,     totalSoldToken,
tokensLeft);
        totalSoldToken = icoTotalAmount;
    }



    /// @notice freeze unfrozenAmount
    function unfrozenTokens() public onlyOwner whenNotPaused {
        require(frozenSupply >= 0);
```

```
    if (contributorsCurrentPeriod < CONTRIBUTORS_PERIODS) {
        unfrozenContributorsTokens();
        unfrozenCompanyTokens();
    } else {
        unfrozenCompanyTokens();
    }
}

// unfrozen contributors token year by year
function unfrozenContributorsTokens() internal {
    require(contributorsCurrentPeriod < CONTRIBUTORS_PERIODS);
    uint256 contributortimeShouldPassBy = contributorsCurrentPeriod * (minutesOneYear);
    TimePassBy("contributortimeShouldPassBy", contributortimeShouldPassBy);
    uint256 contributorsTimePassBy = time() - initDate;
    TimePassBy("contributortimePassBy", contributorsTimePassBy);

    contributorsCurrentPeriod = contributorsCurrentPeriod + 1;
    require(contributorsTimePassBy >= contributortimeShouldPassBy);
    frozenSupply = frozenSupply.sub(contributorsPeriodsElapsed);
    balanceOf[contributorsAddress] = balanceOf[contributorsAddress].add(contributorsPeriodsElapsed);
    LogUnfrozenTokens("contributors", contributorsAddress, contributorsPeriodsElapsed);
}

// unfrozen company token year by year
function unfrozenCompanyTokens() internal {
    require(companyCurrentPeriod < COMPANY_PERIODS);
    uint256 companytimeShouldPassBy = companyCurrentPeriod * (minutesOneYear);
    TimePassBy("CompanytimeShouldPassBy", companytimeShouldPassBy);
    uint256 companytimePassBy = time() - initDate;
    TimePassBy("CompanytimePassBy", companytimePassBy);

    require(companytimePassBy >= companytimeShouldPassBy);
    companyCurrentPeriod = companyCurrentPeriod + 1;
    frozenSupply = frozenSupply.sub(companyPeriodsElapsed);
    balanceOf[companyAddress] = balanceOf[companyAddress].add(companyPeriodsElapsed);
    LogUnfrozenTokens("company", companyAddress, companyPeriodsElapsed);
}

// fallback function - do not allow any eth transfers to this contract
function() external {
    revert();
}
```

```
}
```

慢雾科技
slow mist

官方网址

www.slowmist.com

电子邮箱

team@slowmist.com

微信公众号