链安科技

Blockchain Security

智能合约安全审计报告

链安科技 Blockchain Security

审计合约名称：

CZR

审计合约地址：

0x0223fc70574214f65813fe336d870ac47e147fae

审计合约链接地址：

https://etherscan.io/address/0x0223fc70574214f65813fe336d870ac47e147fae#code

审计合约开始日期：2018.5.31

审计合约完成日期：2018.5.31

审计团队：成都链安科技有限公司

审计类型及结果：

序号	审计类型	审计子项	审计结果
1	代码规范审计	基本格式规范审计	通过
		命名约束规范审计	通过
		权限声明规范审计	通过
		代码设计规范审计	建议
		Gas 消耗审计	通过
		安全函数使用审计	建议
		Fallback 函数使用审计	建议
2	函数调用审计	函数调用权限审计	通过
		call 调用安全审计	通过
		Delegatecall 调用安全审计	通过
		自杀函数调用权限安全审计	通过
3	整型溢出审计		建议
4	可重入攻击审计		通过
5	异常可达状态审计		通过
6	多签名钱包审计		通过

7	执行顺序依赖审计		通过
8	时间戳依赖审计		通过
9	Tx.origin 漏洞审计		通过

备注：审计意见及建议见代码注释

免责声明：本次审计仅针对审计结果表中给定的审计类型范围，其他未知安全漏洞不在本次审计责任范围之内。成都链安科技仅根据本报告出具前已经发生或存在的漏洞出具本报告，并就此承担相应责任。对于出具以后发生或存在的新的攻击或漏洞，成都链安科技无法判断其智能合约的安全状况，亦不对此承担责任。本报告所作的安全审计分析及其他内容，仅基于合约提供者截至本报告出具时向成都链安科技提供的文件和资料，并且文件和资料不存在缺失、被篡改、删减或隐瞒的情形。如已提供的文件和资料信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符的，成都链安科技对由此而导致的损失和不利影响不承担任何责任。

合约源代码审计注释：

```
 pragma solidity ^0.4.16;


// 成都链安 // 合约不存在条件竞争问题

// 成都链安 // 使用了大量 ERC20 标准模块，值得称赞的做法



contract owned {
    address public owner;


    function owned() public {
        owner = msg.sender;
    }


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

```
  function transferOwnership(address newOwner) onlyOwner public {
    // 成都链安 // require(newOwner != address(0));
    // 成都链安 // 建议进行目标地址不为 0 的检查，避免用户失误导致合约控制权彻底丢失
    owner = newOwner;
  }
}

interface tokenRecipient { function receiveApproval(address _from, uint256 _value, address _token, bytes _extraData) public; }

contract TokenERC20 {
  // Public variables of the token
  string public name;
  string public symbol;
  uint8 public decimals = 18;
  // 18 decimals is the strongly suggested default, avoid changing it
  uint256 public totalSupply;

  // This creates an array with all balances
  mapping (address => uint256) public balanceOf;
  mapping (address => mapping (address => uint256)) public allowance;

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

  // This notifies clients about the amount burnt
  event Burn(address indexed from, uint256 value);

  /**
```

```
 * Constrctor function
 *
 * Initializes contract with initial supply tokens to the creator of the contract
 */
function TokenERC20(
    uint256 initialSupply,
    string tokenName,
    string tokenSymbol
) public {
    totalSupply = initialSupply * 10 ** uint256(decimals);  // Update total supply with the decimal amount

    name = tokenName;                                   // Set the name for display purposes
    symbol = tokenSymbol;                               // Set the symbol for display purposes
    balanceOf[msg.sender] = totalSupply;                // Give the creator all initial tokens
}

/**
 * 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);   // 成都链安 // 避免用户失误导致 Token 转丢，值得称赞的写法
    // Check if the sender has enough
    require(balanceOf[_from] >= _value);
    // Check for overflows
    require(balanceOf[_to] + _value > balanceOf[_to]);
    // Save this for an assertion in the future
    uint previousBalances = balanceOf[_from] + balanceOf[_to];
    // Subtract from the sender
```

```
        balanceOf[_from] -= _value;

        // Add the same to the recipient

        balanceOf[_to] += _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] + balanceOf[_to] == previousBalances);  // 成都链安 // 这类设计很好，避免溢出导致 Token 凭空
增加

    }


    /**

     * 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);    // 成都链安 // 很独特的写法，提高复用率的同时，保证交易安全性

    }


    /**

     * Transfer tokens from other address

     *

     * Send `_value` tokens to `_to` in behalf of `_from`

     *

     * @param _from The address of the sender

     * @param _to The address of the recipient

     * @param _value the amount to send
```

```
 */
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success) {
   require(_value <= allowance[_from][msg.sender]);     // Check allowance
   allowance[_from][msg.sender] -= _value;
   _transfer(_from, _to, _value);
   return true;
}

/**
 * Set allowance for other address
 *
 * Allows `_spender` to spend no more than `_value` tokens in your behalf
 *
 * @param _spender The address authorized to spend
 * @param _value the max amount they can spend
 */
function approve(address _spender, uint256 _value) public
   returns (bool success) {
   allowance[msg.sender][_spender] = _value;
   return true;
}

/**
 * Set allowance for other address and notify
 *
 * Allows `_spender` to spend no more than `_value` tokens in your behalf, and then ping the contract about it
 *
 * @param _spender The address authorized to spend
 * @param _value the max amount they can spend
 * @param _extraData some extra information to send to the approved contract
 */
function approveAndCall(address _spender, uint256 _value, bytes _extraData)
    public
    returns (bool success) {
    tokenRecipient spender = tokenRecipient(_spender);
    if (approve(_spender, _value)) {
        spender.receiveApproval(msg.sender, _value, this, _extraData);
        return true;
    }
}

/**
 * Destroy tokens
 *
 * Remove `_value` tokens from the system irreversibly
 *
 * @param _value the amount of money to burn
 */
function burn(uint256 _value) public returns (bool success) {
    require(balanceOf[msg.sender] >= _value);   // Check if the sender has enough
    balanceOf[msg.sender] -= _value;            // Subtract from the sender
    totalSupply -= _value;                      // Updates totalSupply
    Burn(msg.sender, _value);
    return true;
}

/**
 * Destroy tokens from other account
 *
 * Remove `_value` tokens from the system irreversibly on behalf of `_from`.
```

```
     *
     * @param _from the address of the sender
     * @param _value the amount of money to burn
     */
    function burnFrom(address _from, uint256 _value) public returns (bool success) {
        require(balanceOf[_from] >= _value);                // Check if the targeted balance is enough
        require(_value <= allowance[_from][msg.sender]);    // Check allowance
        balanceOf[_from] -= _value;                         // Subtract from the targeted balance
        allowance[_from][msg.sender] -= _value;             // Subtract from the sender's allowance
        totalSupply -= _value;                              // Update totalSupply
        Burn(_from, _value);
        return true;
    }
}

/******************************************/
/*       ADVANCED TOKEN STARTS HERE       */
/******************************************/

contract CZRToken is owned, TokenERC20 {

    mapping (address => bool) public frozenAccount;

    /* This generates a public event on the blockchain that will notify clients */
    event FrozenFunds(address target, bool frozen);

    /* Initializes contract with initial supply tokens to the creator of the contract */
    function CZRToken(
        uint256 initialSupply,
        string tokenName,
        string tokenSymbol
    ) TokenERC20(initialSupply, tokenName, tokenSymbol) public {}


    /* Internal transfer, only can be called by this contract */
    function _transfer(address _from, address _to, uint _value) internal {
        require (_to != 0x0);                        // Prevent transfer to 0x0 address. Use burn() instead
```

// 成都链安 // 这类检查很好，避免用户失误导致 Token 转丢

```
    require (balanceOf[_from] >= _value);            // Check if the sender has enough
    require (balanceOf[_to] + _value > balanceOf[_to]); // Check for overflows
    require(!frozenAccount[_from]);                  // Check if sender is frozen
    require(!frozenAccount[_to]);                    // Check if recipient is frozen
    balanceOf[_from] -= _value;                      // Subtract from the sender
    balanceOf[_to] += _value;                        // Add the same to the recipient
    Transfer(_from, _to, _value);
}


/// @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) onlyOwner public {
    frozenAccount[target] = freeze;
    FrozenFunds(target, freeze);
}
```

// 成都链安 // 独特的设计。这里需要再次确认一个需求逻辑，冻结、释放只会影响拥有者交易和接收 Token，并不会影响拥有者对所拥有的 Token 销毁，授权等。

```
/// Init balances from old CNC chain
/// @param addrs Address array
/// @param balances balance array
```

```
  function init(address[] addrs, uint256[] balances) onlyOwner public {
    require(addrs.length == balances.length);
    uint totalValue;
    for (uint i = 0; i < addrs.length; i++) {
      if (balanceOf[addrs[i]] == 0) {
        var value = balances[i];
        balanceOf[addrs[i]] += value;  // 成都链安 // 存在溢出可能，错误地输入溢出数据会破坏整个合约逻辑。建议计算完全使用 Safemath 库 ，不过只要 init 传入原始链正常余额 ，不会影响合约安全使用。只可能被发币者利用，而非黑客
        Transfer(owner, addrs[i], value);
        totalValue += value;  // 成都链安 // 存在溢出可能，错误地输入溢出数据会破坏整个合约逻辑。建议计算完全使用 Safemath 库 ，不过只要 init 传入原始链正常余额 ，不会影响合约安全使用。只可能被发币者利用，而非黑客
      }
    }
    balanceOf[owner] -= totalValue;   // 成都链安 // 存在溢出可能，错误地输入溢出数据会破坏整个合约逻辑。建议计算完全使用 Safemath 库 ，不过只要 init 传入原始链正常余额 ，不会影响合约安全使用。只可能被发币者利用，而非黑客
  }
}
```

// 成都链安 // 建议主合约继承 Pausable ERC20 标准模块，当出现重大异常时可以暂停所有交易

// 成都链安 // function () public payable{ revert(); }

// 成都链安 // 建议增加这样的 fallback 函数，如果没有 fallback 函数用户向该合约地址转入 ETH 则无法退回，最终导致丢失所转的 ETH

链安科技

Blockchain Security

官方网址

http://lianantech.com

电子邮箱

vaas@lianantech.com

微信公众号