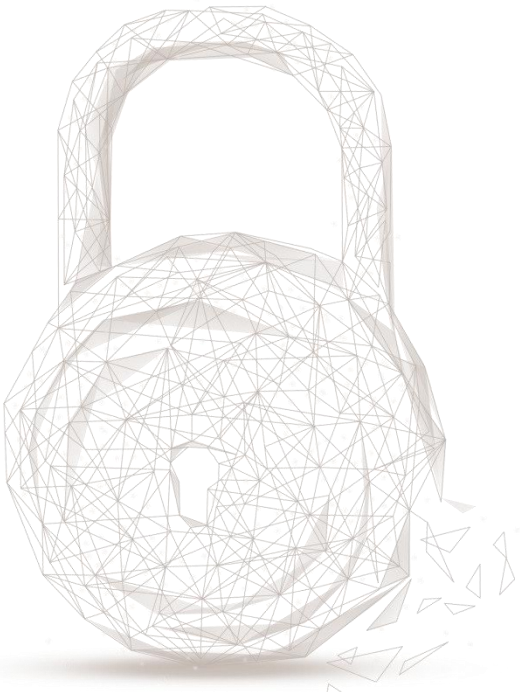
成都链安
BEOSIN

智能合约安全审计报告



审计编号：202009240945

审计合约名称：

WBCD

审计合约地址：

0x6A6d430573D3F070AEAb97b3A189d698eA130454

审计合约链接地址：

https://etherscan.io/address/0x6a6d430573d3f070aeab97b3a189d698ea130454#code

合约审计开始日期：2020.09.21

合约审计完成日期：2020.09.24

审计结果：通过（优）

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	ERC20 Token 标准规范审计	通过
		编译器版本安全审计	通过
		可见性规范审计	通过
		gas 消耗审计	通过
		SafeMath 功能审计	通过
		fallback 函数使用审计	通过
		tx.origin 使用审计	通过
		弃用项审计	通过
		冗余代码审计	通过
		变量覆盖审计	通过
2	函数调用审计	函数调用权限审计	通过
		call/delegatecall 安全审计	通过
		返回值安全审计	通过
		自毁函数安全审计	通过
3	业务安全审计	owner 权限审计	通过
		业务逻辑审计	通过
		业务实现审计	通过
4	整型溢出审计	-	通过
5	可重入攻击审计	-	通过
6	异常可达状态审计	-	通过
7	交易顺序依赖审计	-	通过
8	块参数依赖审计	-	通过
9	伪随机数生成审计	-	通过

10	拒绝服务攻击审计	-	通过
11	代币锁仓审计	-	无锁仓
12	假充值审计	-	通过
13	event 安全审计	-	通过

备注：审计意见及建议请见代码注释。

免责声明：本次审计仅针对本报告载明的审计类型及结果表中给定的审计类型范围进行审计，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存在或发生的攻击或漏洞出具本报告，对于出具以后存在或发生的新的攻击或漏洞，成都链安科技无法判断其对智能合约安全状况可能的影响，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，仅基于合约提供者在本报告出具前已向成都链安科技提供的文件和资料，且该部分文件和资料不存在任何缺失、被篡改、删减或隐瞒的前提下作出的；如提供的文件和资料存在信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符等情况或提供文件和资料在本报告出具后发生任何变动的，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。成都链安科技出具的本审计报告系根据合约提供者提供的文件和资料依靠成都链安科技现掌握的技术而作出的，由于任何机构均存在技术的局限性，成都链安科技作出的本审计报告仍存在无法完整检测出全部风险的可能性，成都链安科技对由此产生的损失不承担任何责任。

本声明最终解释权归成都链安科技所有。

审计结果说明：

本公司采用形式化验证、静态分析、动态分析、典型案例测试和人工审核的方式对智能合约WBCD的代码规范性、安全性以及业务逻辑三个方面进行多维度全面的安全审计。**经审计，WBCD合约通过所有检测项，合约审计结果为通过(优)，合约可正常使用。**以下为本合约基本信息。

1、代币基本信息

Token name	Wrapped Bitcoin Diamond
Token symbol	WBCD
decimals	7
totalSupply	100万（可增发，上限2.1亿）
Token type	ERC20

表1 代币基本信息

2、代币锁仓信息

无锁仓

合约源代码审计注释：

```
/**
 *Submitted for verification at Etherscan.io on 2020-09-23
*/

pragma solidity ^0.5.17; // 成都链安 // 建议固定编译器版本
// ----------------------------------------------------------------------------

// ERC Token Standard #20 Interface

// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md

// ----------------------------------------------------------------------------

contract ERC20Interface {
  // 成都链安 // 定义 ERC20 Token 标准要求的接口函数
  function totalSupply() public view returns (uint);
  function balanceOf(address tokenOwner) public view returns (uint balance);
  function allowance(address tokenOwner, address spender) public view returns (uint
remaining);
  function transfer(address to, uint tokens) public returns (bool success);
  function approve(address spender, uint tokens) public returns (bool success);
  function transferFrom(address from, address to, uint tokens) public returns (bool
success);
  // 成都链安 // 声明代币转账事件
  event Transfer(address indexed from, address indexed to, uint tokens);
  // 成都链安 // 声明代币授权事件
  event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}

// ----------------------------------------------------------------------------
// Safe maths
// ----------------------------------------------------------------------------
// 成都链安 // SafeMath 库定义以下函数用于安全数学运算
library SafeMath {
  function add(uint a, uint b) internal pure returns (uint c) {
    c = a + b;
    require(c >= a);
  }
  function sub(uint a, uint b) internal pure returns (uint c) {
    require(b <= a);
    c = a - b;
  }
  function mul(uint a, uint b) internal pure returns (uint c) {
    c = a * b;
    require(a == 0 || c / a == b); // the same as: if (a !=0 && c / a != b) {throw;}
  }
  function div(uint a, uint b) internal pure returns (uint c) {
```

```
    require(b > 0);
    c = a / b;
  }
}

// ----------------------------------------------------------------------------
// Ownable Contract
// ----------------------------------------------------------------------------
contract Ownable {
  address public owner; // 成都链安 // 声明变量，存储合约管理员地址
  event OwnershipTransferred(address indexed previousOwner, address indexed newOwner); // 成都链安 // 声明合约管理权限转移事件

  // The Ownable constructor sets the original `owner` of the contract to the sender account
  // 成都链安 // Owanble 合约构造函数，初始化地址 msg.sender 为本合约的管理员
  constructor() public {
    owner = msg.sender;
  }
  // 成都链安 // 函数修饰器，要求被修饰函数的调用者必须是合约管理员
  modifier onlyOwner() {
    require(msg.sender == owner);
    _;
  }

  // Allows the current owner to transfer control of the contract to a newOwner.
  // @param newOwner The address to transfer ownership to.
  // 成都链安 // 合约管理权限转移函数，转移合约管理权限至地址 newOwner
  function transferOwnership(address newOwner) public onlyOwner {
    require(newOwner != address(0)); // 成都链安 // newOwner 非零地址检查
    emit OwnershipTransferred(owner, newOwner); // 成都链安 // 触发合约管理权限转移事件
    owner = newOwner; // 成都链安 // 设置地址 newOwner 为新的合约管理员
  }
}


// ----------------------------------------------------------------------------
// ERC20 Token, with the addition of symbol, name and decimals and an
// initial fixed supply
// ----------------------------------------------------------------------------
contract WBCDToken is ERC20Interface, Ownable{
  using SafeMath for uint; // 成都链安 // 引入 SafeMath 安全数学运算库，避免数学运算整型溢出
  // 成都链安 // 声明变量，存储代币基本信息
  string public symbol;
  string public  name;
  uint8 public decimals;
  uint _totalSupply;

  mapping(address => uint) balances; // 成都链安 // 声明变量，存储指定地址的代币余额
  mapping(address => mapping(address => uint)) allowed; // 成都链安 // 声明变量，存储指定地
```

址间的授权值

```
    event IncrementSupply(address indexed owner, uint256 increments); // 成都链安 // 声明代币增发事件
    event BurnTokens(address indexed owner, uint256 amount); // 成都链安 // 声明代币销毁事件

  // ------------------------------------------------------------------------
  // Constructor
  // ------------------------------------------------------------------------
  // 成都链安 // 构造函数，初始化代币基本信息、代币总量和合约管理员
  constructor() public {
    symbol = "WBCD";
    name = "Wrapped Bitcoin Diamond";
    decimals = 7;
    _totalSupply = 1000000 * 10**uint(decimals);
    balances[msg.sender] = _totalSupply; // 成都链安 // 发送全部初始代币至地址 msg.sender
    emit Transfer(address(0), msg.sender, _totalSupply); // 成都链安 // 触发代币转账事件
  }

  // ------------------------------------------------------------------------
  // Total supply
  // ------------------------------------------------------------------------
  function totalSupply() public view returns (uint) {
    return _totalSupply;
  }

  // ------------------------------------------------------------------------
  // Get the token balance for account `tokenOwner`
  // ------------------------------------------------------------------------
  function balanceOf(address tokenOwner) public view returns (uint balance) {
    return balances[tokenOwner];
  }

  // ------------------------------------------------------------------------
  // Transfer the balance from token owner's account to `to` account
  // - Owner's account must have sufficient balance to transfer
  // - 0 value transfers are allowed
  // ------------------------------------------------------------------------
  function transfer(address to, uint tokens) public returns (bool success) {
    require(to != address(0), "to address is a zero address"); // 成都链安 // to 非零地址检查
    balances[msg.sender] = balances[msg.sender].sub(tokens); // 成都链安 // 减少函数调用者 msg.sender 的代币余额
    balances[to] = balances[to].add(tokens); // 成都链安 // 增加转账目标地址 to 的代币余额
    emit Transfer(msg.sender, to, tokens); // 成都链安 // 触发代币转账事件
    return true;
  }
  // ------------------------------------------------------------------------
  // Token owner can approve for `spender` to transferFrom(...) `tokens`
  // from the token owner's account
```

```
  //
  // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
  // recommends that there are no checks for the approval double-spend attack
  // as this should be implemented in user interfaces
  // ------------------------------------------------------------------------
  // 成都链安 // 用户调用该函数修改授权值时，可能导致多重授权；建议用户先重置授权值为 0，再进行新的授权
  function approve(address spender, uint tokens) public returns (bool success) {
    require(spender != address(0), "spender address is a zero address"); // 成都链安 // spender 非 0 地址检查
    allowed[msg.sender][spender] = tokens; // 成都链安 // 设置 msg.sender 对 spender 的授权值为 tokens
    emit Approval(msg.sender, spender, tokens); // 成都链安 // 触发 Approval 事件
    return true;
  }

  // ------------------------------------------------------------------------
  // Transfer `tokens` from the `from` account to the `to` account
  //
  // The calling account must already have sufficient tokens approve(...)-d
  // for spending from the `from` account and
  // - From account must have sufficient balance to transfer
  // - Spender must have sufficient allowance to transfer
  // - 0 value transfers are allowed
  // ------------------------------------------------------------------------
  function transferFrom(address from, address to, uint tokens) public returns (bool success) {
    require(to != address(0), "to address is a zero address"); // 成都链安 // to 非零地址检查
    balances[from] = balances[from].sub(tokens); // 成都链安 // 减少转账源地址 from 的代币余额
    allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens); // 成都链安 // 减少转账源地址 from 对函数调用者 msg.sender 的授权值
    balances[to] = balances[to].add(tokens); // 成都链安 // 增加转账目标地址 to 的代币余额
    emit Transfer(from, to, tokens); // 成都链安 // 触发代币转账事件
    return true;
  }

  // ------------------------------------------------------------------------
  // Returns the amount of tokens approved by the owner that can be
  // transferred to the spender's account
  // ------------------------------------------------------------------------
  function allowance(address tokenOwner, address spender) public view returns (uint remaining) {
    return allowed[tokenOwner][spender];
  }

  // ------------------------------------------------------------------------
  // Increment supply
```

```
  // ------------------------------------------------------------------------
  // 成都链安 // 增发指定数量代币，仅合约管理员可调用；总量上限接近 2.1 亿
  function incrementSupply(uint256 increments) public onlyOwner {
    require(increments > 0); // 成都链安 // 增发的数量大于 0 检查
    require(_totalSupply.add(increments) < 210000000 * 10**uint(decimals)); // 成都链安 //
要求增发后代币总量小于 2.1 亿
    uint256 curTotalSupply = _totalSupply; // 成都链安 // 获取代币总量
    uint256 previousBalance = balanceOf(msg.sender); // 成都链安 // 获取 msg.sender 余额
    _totalSupply = curTotalSupply.add(increments); // 成都链安 // 增加代币总量
    balances[msg.sender] = previousBalance.add(increments); // 成都链安 // 增加 msg.sender 余
额
    emit IncrementSupply(msg.sender, increments); // 成都链安 // 触发 IncrementSupply 事件
    emit Transfer(address(0), msg.sender, increments); // 成都链安 // 触发 Transfer 事件
  }

  // ------------------------------------------------------------------------
  // Burns `amount` tokens from `owner`
  // @param amount The quantity of tokens being burned
  // @return True if the tokens are burned correctly
  // ------------------------------------------------------------------------
  // 成都链安 // 销毁指定数量代币，仅合约管理员可以调用
  function burnTokens(uint256 amount) public onlyOwner returns (bool) {
    require(amount > 0); // 成都链安 // 销毁的数量大于 0 检查
    uint256 curTotalSupply = _totalSupply; // 成都链安 // 获取代币总量
    require(curTotalSupply >= amount); // 成都链安 // 代币总量大于等于销毁的数量检查
    uint256 previousBalanceTo = balanceOf(msg.sender); // 成都链安 // 获取 msg.sender 余额
    require(previousBalanceTo >= amount); // 成都链安 // msg.sender 余额大于等于销毁数量检查
    _totalSupply = curTotalSupply.sub(amount); // 成都链安 // 减少代币总量
    balances[msg.sender] = previousBalanceTo.sub(amount); // 成都链安 // 减少 msg.sender 余额
    emit BurnTokens(msg.sender, amount); // 成都链安 // 触发 BurnTokens 事件
    emit Transfer(msg.sender, address(0), amount); // 成都链安 // 触发 Transfer 事件
    return true;
  }
}
// 成都链安 // 建议主合约继承 Pausable 模块，当出现重大异常时 owner 可以暂停所有交易
```

成都链安
BEOSIN

官方网址

https://lianantech.com

电子邮箱

vaas@lianantech.com

微信公众号