智能合约审计报告

降维安全

JohnWick Sec

守护价值互联网

北京互联共识科技有限公司

二零一八年七月

降维安全实验室于 2018 年 7 月 16 日 收到 BTMC（公司/团队）的 MinerCoin (BTMC) 项目智能合约源代码安全审计需求。

项目名称：MinerCoin (BTMC)

合约地址：

https://etherscan.io/address/0x4a8f44be523580a11cdb20e2c7c470adf44ec9bb#code

审计编号：201807121

审计项目及结果：

(其他未知安全漏洞和以太坊设计缺陷不包含在本次审计责任范围内)

审计大类	审计子类	审计结果（通过或未通过）
溢出审计	-	通过
条件竞争	-	通过
访问控制	-	通过
拒绝服务	-	通过
Gas 优化	-	通过
程序设计	编译器版本	通过
	随机数生成	通过
	硬编码地址审计	通过
	回退函数使用	通过
	内部函数调用绕过	通过
	其他显性逻辑错误	未通过
特色服务	代码格式规范化	通过
	模糊测试结果	通过

审计结果：通过

审计日期：20180716

审计团队：降维安全实验室

审计详情如下：

//John Wick：使用 SafeMath 函数库，符合推荐做法。

//John Wick：approve 条件竞争【低危】 ，approve 函数未确认许可金额为 0，建议 allowed 不为 0 时，禁止用户使用此方法 341L
建议增加 require((_value == 0) || (allowed[msg.sender][_spender] == 0));
//John Wick:时间戳依赖【低危】，时间戳可能受到矿工控制，推荐使用区块高度计算时间

```
pragma solidity ^0.4.13;


contract ERC20Basic {
  uint256 public totalSupply;
  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 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 Ownable {
  address public owner;


  event   OwnershipTransferred(address   indexed   previousOwner,   address
```

```
indexed newOwner);


  /**
   * @dev The Ownable constructor sets the original `owner` of the contract
to the sender
   * account.
   */
  function Ownable() 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 transfer control of the contract to
a newOwner.
   * @param newOwner The address to transfer ownership to.
   */
  function transferOwnership(address newOwner) public onlyOwner {
    require(newOwner != address(0));
    OwnershipTransferred(owner, newOwner);
    owner = newOwner;
  }

}

library SafeMath {
  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;
  }
```

```
  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;
  }

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

  function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    assert(c >= a);
    return c;
  }
}

contract Pausable is Ownable {
  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() onlyOwner whenNotPaused public {
    paused = true;
    Pause();
  }

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


contract BTMC is ERC20,Ownable,Pausable{
    using SafeMath for uint256;

    //the base info of the token
    string public constant name="MinerCoin";
    string public constant symbol="BTMC";
    string public constant version = "1.0";
    uint256 public constant decimals = 18;

    //1 亿团队持有
    uint256 public constant INIT_SUPPLY=100000000*10**decimals;

    //挖矿 5 亿（代币阶段先不生成）
    uint256 public constant MINING_SUPPLY=500000000*10**decimals;


    //公募 2 亿
    uint256 public constant MAX_FUNDING_SUPPLY=200000000*10**decimals;

    //团队锁定 2 亿
    uint256 public constant TEAM_KEEPING=200000000*10**decimals;

    //总发行 10 亿
    uint256                         public                         constant
MAX_SUPPLY=INIT_SUPPLY+MINING_SUPPLY+MAX_FUNDING_SUPPLY+TEAM_KEEPING;

    //公募参数
```

```
//已经公募量
uint256 public totalFundingSupply;
uint256 public startTime;
uint256 public endTime;
uint256 public rate;

//团队每次解禁
uint256 public constant TEAM_UNFREEZE=40000000*10**decimals;
bool public hasOneStepWithdraw;
bool public hasTwoStepWithdraw;
bool public hasThreeStepWithdraw;
bool public hasFourStepWithdraw;
bool public hasFiveStepWithdraw;



//ERC20 的余额
mapping(address => uint256) balances;
mapping (address => mapping (address => uint256)) allowed;

function BTMC(){
    totalSupply=INIT_SUPPLY;
    balances[msg.sender] = INIT_SUPPLY;
    Transfer(0x0, msg.sender, INIT_SUPPLY);
    totalFundingSupply = 0;

    //20180423 235959
    startTime=1524499199;
    //20180515 000000
    endTime=1526313600;
    rate=5000;

    hasOneStepWithdraw=false;
    hasTwoStepWithdraw=false;
    hasThreeStepWithdraw=false;
    hasFourStepWithdraw=false;
    hasFiveStepWithdraw=false;




}

event CreateBTMC(address indexed _to, uint256 _value);
```

```
    modifier notReachTotalSupply(uint256 _value,uint256 _rate){
        assert(MAX_SUPPLY>=totalSupply.add(_value.mul(_rate)));
        _;
    }

    modifier notReachFundingSupply(uint256 _value,uint256 _rate){

    assert(MAX_FUNDING_SUPPLY>=totalFundingSupply.add(_value.mul(_rate)))
;
        _;
    }
    modifier assertFalse(bool withdrawStatus){
        assert(!withdrawStatus);
        _;
    }

    modifier notBeforeTime(uint256 targetTime){
        assert(now>targetTime);
        _;
    }

    modifier notAfterTime(uint256 targetTime){
        assert(now<=targetTime);
        _;
    }


    //owner 有权限提取账户中的 eth
    function etherProceeds() external
        onlyOwner

    {
        if(!msg.sender.send(this.balance)) revert();
    }


    //代币分发函数，内部使用
    function processFunding(address receiver,uint256 _value,uint256 _rate)
internal
        notReachTotalSupply(_value,_rate)
    {
        uint256 amount=_value.mul(_rate);
        totalSupply=totalSupply.add(amount);
        balances[receiver] +=amount;
```

```
        CreateBTMC(receiver,amount);
        Transfer(0x0, receiver, amount);
    }

    function funding (address receiver,uint256 _value,uint256 _rate)
whenNotPaused internal
        notReachFundingSupply(_value,_rate)
    {
        processFunding(receiver,_value,_rate);
        uint256 amount=_value.mul(_rate);
        totalFundingSupply = totalFundingSupply.add(amount);
    }


    function () payable external
        notBeforeTime(startTime)
        notAfterTime(endTime)
    {
            funding(msg.sender,msg.value,rate);
    }


    //20200423 000000 可提
    function withdrawForOneStep() external
        onlyOwner
        assertFalse(hasOneStepWithdraw)
        notBeforeTime(1587571200)
    {
        processFunding(msg.sender,TEAM_UNFREEZE,1);
        //标记团队已提现
        hasOneStepWithdraw = true;
    }

    //20201023 000000
    function withdrawForTwoStep() external
        onlyOwner
        assertFalse(hasTwoStepWithdraw)
        notBeforeTime(1603382400)
    {
        processFunding(msg.sender,TEAM_UNFREEZE,1);
        //标记团队已提现
        hasTwoStepWithdraw = true;
    }

    //20210423 000000
```

```
    function withdrawForThreeStep() external
        onlyOwner
        assertFalse(hasThreeStepWithdraw)
        notBeforeTime(1619107200)
    {
        processFunding(msg.sender,TEAM_UNFREEZE,1);
        //标记团队已提现
        hasThreeStepWithdraw = true;
    }

    //20211023 000000
    function withdrawForFourStep() external
        onlyOwner
        assertFalse(hasFourStepWithdraw)
        notBeforeTime(1634918400)
    {
        processFunding(msg.sender,TEAM_UNFREEZE,1);
        //标记团队已提现
        hasFourStepWithdraw = true;
    }

    //20220423 000000
    function withdrawForFiveStep() external
        onlyOwner
        assertFalse(hasFiveStepWithdraw)
        notBeforeTime(1650643200)
    {
        processFunding(msg.sender,TEAM_UNFREEZE,1);
        //标记团队已提现
        hasFiveStepWithdraw = true;
    }


    function transfer(address _to, uint256 _value) whenNotPaused public
returns (bool)
    {
        require(_to != address(0));
        // SafeMath.sub will throw if there is not enough balance.
        balances[msg.sender] = balances[msg.sender].sub(_value);
        balances[_to] = balances[_to].add(_value);
        Transfer(msg.sender, _to, _value);
        return true;
    }

    function balanceOf(address _owner) public constant returns (uint256
```

```
balance)
    {
        return balances[_owner];
    }


    function transferFrom(address _from, address _to, uint256 _value)
whenNotPaused public returns (bool)
    {
        require(_to != address(0));
        uint256 _allowance = allowed[_from][msg.sender];
        balances[_from] = balances[_from].sub(_value);
        balances[_to] = balances[_to].add(_value);
        allowed[_from][msg.sender] = _allowance.sub(_value);
        Transfer(_from, _to, _value);
        return true;
    }

    function approve(address _spender, uint256 _value) whenNotPaused public
returns (bool)
    {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true;
    }

    function allowance(address _owner, address _spender) public constant
returns (uint256 remaining)
    {
        return allowed[_owner][_spender];
    }


    function setupFundingRate(uint256 _rate) external
        onlyOwner
    {
        rate=_rate;
    }

    function   setupFundingTime(uint256   _startTime,uint256   _endTime)
external
        onlyOwner
    {
        startTime=_startTime;
        endTime=_endTime;
```

```
    }

}
```