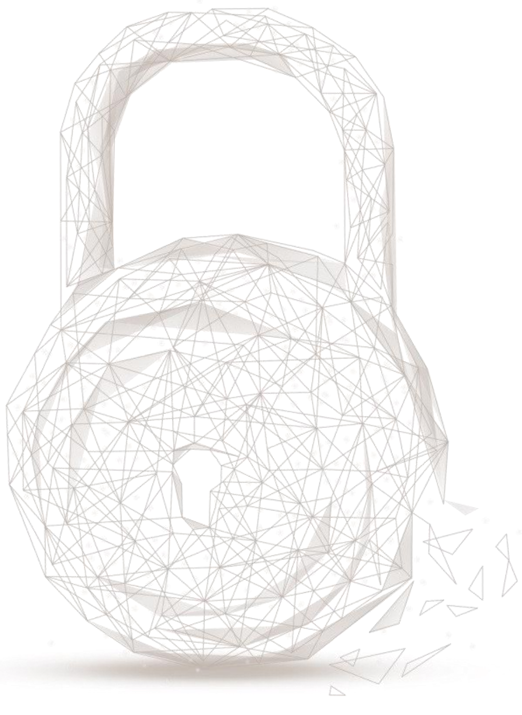
链安科技

Blockchain Security

智能合约安全审计报告



审计编号：201807101447

审计合约名称：

YouDeal Token (YD)

审计合约地址：

0x78FcEca5bf5EC79C23effece97Ae758665BA4f55

审计合约链接地址：

https://etherscan.io/address/0x78fceca5bf5ec79c23effece97ae758665ba4f55#code

审计合约开始日期：2018.06.29

审计合约完成日期：2018.07.10

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
		可见性规范审计	通过
		gas 消耗审计	通过
		SafeMath 函数使用审计	通过
		fallback 函数使用审计	通过
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	-	通过
4	可重入攻击审计	-	通过
5	异常可达状态审计	-	通过
6	交易顺序依赖审计	-	通过

7	时间戳依赖审计	-	通过
8	tx.origin 使用审计	-	通过
9	代币库锁仓审计	-	通过

备注：审计意见及建议见代码注释

合约源代码审计注释：

```
pragma solidity ^0.4.21;// 成都链安 // 建议固定编译器版本

/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
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;
  }

}

/**
 * @title ERC20Basic
 * @dev Simpler version of ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/179
 */
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);
}


/**
 * @title ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 */
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);
```

```
}


/**
 * @title Basic token
 * @dev Basic version of StandardToken, with no allowances.
 */
contract BasicToken is ERC20Basic {
  using SafeMath for uint256;

  mapping(address => uint256) balances;

  /**
  * @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) {
    require(_to != address(0));
    require(_value <= balances[msg.sender]);

    // SafeMath.sub will throw if there is not enough balance.
    balances[msg.sender] = balances[msg.sender].sub(_value);
    balances[_to] = balances[_to].add(_value);
    emit Transfer(msg.sender, _to, _value);
    return true;
  }

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

```
}

/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * @dev https://github.com/ethereum/EIPs/issues/20
 * @dev Based on code by FirstBlood:
https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.sol
 */
contract StandardToken is ERC20, BasicToken {

  mapping (address => mapping (address => uint256)) internal allowed;


  /**
   * @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(_to != address(0));
    require(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

    balances[_from] = balances[_from].sub(_value);
    balances[_to] = balances[_to].add(_value);
    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
    emit Transfer(_from, _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) {
    allowed[msg.sender][_spender] = _value;
    emit Approval(msg.sender, _spender, _value);
    return true;
  }

  /**
   * @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 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 increaseApproval(address _spender, uint _addedValue) public returns (bool) {
    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 decreaseApproval(address _spender, uint _subtractedValue) public returns (bool) {
    uint oldValue = allowed[msg.sender][_spender];
    if (_subtractedValue > oldValue) {
      allowed[msg.sender][_spender] = 0;
    } else {
      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
    }
    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    return true;
  }

}

/**
 * @title Ownable
 * @dev The Ownable contract has an owner address, and provides basic authorization control
 * functions, this simplifies the implementation of "user permissions".
 */
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));
    emit OwnershipTransferred(owner, newOwner);
    owner = newOwner;
  }

}

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;
    emit Pause();
  }

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

}

contract PausableToken is StandardToken, 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 increaseApproval(address _spender, uint _addedValue) public whenNotPaused
returns (bool success) {
    return super.increaseApproval(_spender, _addedValue);
  }

  function decreaseApproval(address _spender, uint _subtractedValue) public whenNotPaused
returns (bool success) {
    return super.decreaseApproval(_spender, _subtractedValue);
  }

}

/**
 * @title YouDeal Token
 * @dev YDChain.
 */
contract YouDealToken is PausableToken {

  string public constant name = "YouDeal Token";
  string public constant symbol = "YD";
  uint8 public constant decimals = 18;

  uint256 private constant TOKEN_UNIT = 10 ** uint256(decimals);
  uint256 private constant INITIAL_SUPPLY = 10500000000 * TOKEN_UNIT;

  uint256 private constant PRIVATE_SALE_SUPPLY = INITIAL_SUPPLY * 25 / 100;  // 25% for
private sale
```

```
  uint256 private constant COMMUNITY_REWARDS_SUPPLY = INITIAL_SUPPLY * 40 / 100;  // 40% for community rewards
  uint256 private constant FOUNDATION_SUPPLY = INITIAL_SUPPLY * 20 / 100;  // 20% for foundation
  // 成都链安 // 锁仓代币总量为代币总发行量的 15%
  uint256 private constant TEAM_SUPPLY = INITIAL_SUPPLY * 15 / 100;  // 15% for founder team

  struct VestingGrant {
        address beneficiary;
        uint256 start;
        uint256 duration; //duration for each release
        uint256 amount; //total grant amount
        uint256 transfered; // transfered amount
        uint8 releaseCount; // schedule release count
  }

  address private constant PRIVAYE_SALE_ADDRESS = 0x65158a7270b58fd9499bE7E95feFBF2169360728; //team vesting  beneficiary address
  address private constant COMMUNITY_REWARDS_ADDRESS = 0xDFE95879606F520CaC6a3546FE2f0d8BBC10A32b; //community rewards wallet address
  address private constant FOUNDATION_ADDRESS = 0xC138e8A6763e78fA0fFAD6c392D01e37CF3fdf27; //foundation wallet address

  VestingGrant teamVestingGrant;

  /**
   * @dev Constructor that gives msg.sender all of existing tokens.
   */
  function YouDealToken() public {
    totalSupply =  INITIAL_SUPPLY;

    balances[PRIVAYE_SALE_ADDRESS] = PRIVATE_SALE_SUPPLY;
    balances[COMMUNITY_REWARDS_ADDRESS] = COMMUNITY_REWARDS_SUPPLY;
    balances[FOUNDATION_ADDRESS] = FOUNDATION_SUPPLY;
    // 成都链安 // 锁仓设置：锁仓代币的解锁账户为合约的创建者账户，合约部署完成后 180 天后开始解锁，每 30 天解锁一次，共计解锁 30 次
    teamVestingGrant = founderGrant(msg.sender, now.add(150 days), (30 days),
TEAM_SUPPLY, 30); // The owner address is reserved for the Team Wallet
  }

  function founderGrant(address _beneficiary, uint256 _start, uint256 _duration, uint256
_amount, uint8 _releaseCount)
    internal pure returns  (VestingGrant) {
      return VestingGrant({ beneficiary : _beneficiary, start: _start, duration:_duration,
amount:_amount, transfered:0, releaseCount:_releaseCount});
  }
  // 成都链安 // 调用内部函数 relaseVestingGrant 解锁代币
  function releaseTeamVested() public onlyOwner {
      relaseVestingGrant(teamVestingGrant);
  }
  // 成都链安 // 计算应解锁代币数量
  function releasableAmount(uint256 time, VestingGrant grant) internal pure returns (uint256) {
      if (grant.amount == grant.transfered) {// 成都链安 // 判断已解锁代币是否等于锁仓代币总数
          return 0;
      }
    if (time < grant.start) {// 成都链安 // 判断是否到达解锁日期
          return 0;
      }
      uint256 amountPerRelease = grant.amount.div(grant.releaseCount);// 成都链安 // 将锁仓代
币总量平均分为 30 份，每 30 天解锁一份
      uint256 amount = amountPerRelease.mul((time.sub(grant.start)).div(grant.duration));// 成
都链安 // 计算在 time 时刻，应解锁代币额度
      if (amount > grant.amount) {
        amount = grant.amount;
      }
      amount = amount.sub(grant.transfered);// 成都链安 // 计算还未发送到解锁账户的代币数量
      return amount;
  }
  // 成都链安 // 将已解锁的代币发送至解锁账户
  function relaseVestingGrant(VestingGrant storage grant) internal {
      uint256 amount = releasableAmount(now, grant);// 成都链安 // 调用内部函数
releasableAmount 计算当前时间应发送至解锁账户代币数量
      require(amount > 0);
      grant.transfered = grant.transfered.add(amount);
      balances[grant.beneficiary] = balances[grant.beneficiary].add(amount);
      emit Transfer(address(0), grant.beneficiary, amount);
  }

}
```

链安科技
Blockchain Security

官方网址

https://lianantech.com

电子邮箱

vaas@lianantech.com

微信公众号