智能合约安全审计报告

慢雾安全团队于 2018-09-14 日，收到 LVECoin 团队对 LVECoin 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：

LVE

合约地址：

LVECoin.sol: 0x428d941e0a014bb5cdeb09bb00bc7b245221bdb0

FoundingTeam.sol: 0x652c60cc42078a923dbf3a19be61c0040bd02c6e

链接地址：

LVECoin.sol:

https://etherscan.io/address/0x428d941e0a014bb5cdeb09bb00bc7b245221bdb0

FoundingTeam.sol:

https://etherscan.io/address/0x652c60cc42078a923dbf3a19be61c0040bd02c6e

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	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
10	未初始化的存储指针	-	通过
11	算术精度误差	-	通过

备注：审计意见及建议见代码注释 **//SlowMist//.....**

审计结果：**通过**

审计编号：0X001809190001

审计日期：2018 年 09 月 19 日

审计团队：慢雾安全团队

总结：此为代币(token)合约，含锁仓(tokenVault)部分，合约使用了 OpenZeppelin 的 SafeMath 安全模块，值得称赞的做法，综合评估合约无风险。

合约源代码如下：

LVECoin.sol

//SlowMist// 合约不存在溢出、条件竞争问题

```
pragma solidity ^0.4.24;


// *----------------------------------------------------------------------*
//       __ _    _______   _________  _____   __
//      / /| |  / / ____/  / ____/ __ \/  _/ | / /
//     / / | | / / __/    / /   / / / // //  |/ /
//    / /__| |/ / /___   / /___/ /_/ // // /|  /
//   /_____/___/_____/   \____/\____/___/_/ |_/
// *----------------------------------------------------------------------*
```

```
/**
 * @title SafeMath
 */
```

//SlowMist// 使用了 OpenZeppelin 的 SafeMath 安全模块，值得称赞的做法

```
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) {
        uint256 c = a / b;
        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 Ownable
 * @ multiSig
 */
contract Ownable {

    // _from: oldOwner _to: newOwner
```

```
event OwnershipTransferred(address indexed _from, address indexed _to);
event SubmitPrps(ProposalType indexed _prpsType);
event SignPrps(uint256 indexed _prpsIdx, ProposalType indexed _prpsType, address indexed _from);

// owner proposal type enum
enum ProposalType {
    freeze,
    unfreeze,
    transferOwner
}
// owner proposal
struct Proposal {
    ProposalType prpsType;
    address fromAddr;
    address toAddr;
    mapping(address => bool) signed;
    bool finalized;
}
// require sign owner number
uint256 public requiredSignNum;
// all owner address
address[] public owners;
// owner proposal list
Proposal[] public proposals;
// is owner mapping
mapping(address => bool) public isOwnerMap;

constructor() public{
}

// is owner
modifier isOwner{
    require(isOwnerMap[msg.sender], "");
    _;
}
// is most owner sign proposal
modifier multiSig(uint256 _prpsIdx) {
    // is more than half(多数决)
    require(signOwnerCount(_prpsIdx) >= requiredSignNum, "");
    // proposal is not finalized
    require(proposals[_prpsIdx].finalized == false, "");
    _;
```

```
    }
    // proposal is exist
    modifier isPrpsExists(uint256 _prpsIdx) {
        require(_prpsIdx >= 0, "");
        require(_prpsIdx < proposals.length, "");
        _;
    }
    modifier checkOwner(address _fromAddr, address _toAddr) {

        require(_toAddr != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas
```

的消耗

```
        require(_toAddr != msg.sender, "");
        require(_fromAddr != msg.sender, "");
        _;
    }
    // is right proposal type
    modifier checkPrpsType(ProposalType _prpsType) {
        require(_prpsType == ProposalType.freeze || _prpsType == ProposalType.unfreeze || _prpsType ==
ProposalType.transferOwner, "");
        _;
    }
    // check already sign prps
    modifier checkSignPrps(uint256 _prpsIdx) {
        // proposal is not finalized
        require(proposals[_prpsIdx].finalized == false, "");
        // owner is not signed
        require(proposals[_prpsIdx].signed[msg.sender] == false, "");
        _;
    }


    // any owner submit not certified proposal
    function submitProposal(ProposalType _prpsType, address _fromAddr, address _toAddr) public isOwner
checkOwner(_fromAddr, _toAddr) checkPrpsType(_prpsType) {
        Proposal memory _proposal;
        _proposal.prpsType = _prpsType;
        _proposal.finalized = false;
        _proposal.fromAddr = _fromAddr;
        _proposal.toAddr = _toAddr;
        proposals.push(_proposal);
        emit SubmitPrps(_prpsType);
```

```
    }

    // owner sign an proposal
    function signProposal(uint256 _prpsIdx) public isOwner isPrpsExists(_prpsIdx) checkSignPrps(_prpsIdx){
        proposals[_prpsIdx].signed[msg.sender] = true;
        emit SignPrps(_prpsIdx, proposals[_prpsIdx].prpsType, msg.sender);
    }

    // get proposal owner sign number(多数决)
    function signOwnerCount(uint256 _prpsIdx) public view isPrpsExists(_prpsIdx) returns(uint256) {
        uint256 signedCount = 0;
        for(uint256 i = 0; i < owners.length; i++) {
            if(proposals[_prpsIdx].signed[owners[i]] == true){
                signedCount++;
            }
        }
        return signedCount;
    }

    // proposal count nums
    function getProposalCount() public view returns(uint256){
        return proposals.length;
    }

    // get proposal sign status info
    function getProposalInfo(uint256 _prpsIdx) public view isPrpsExists(_prpsIdx) returns(ProposalType
_prpsType, uint256 _signedCount, bool _isFinalized, address _fromAddr, address _toAddr){

        Proposal memory _proposal = proposals[_prpsIdx];
        uint256 signCount = signOwnerCount(_prpsIdx);
        return (_proposal.prpsType, signCount, _proposal.finalized, _proposal.fromAddr,
_proposal.toAddr);
    }

    // Transfer owner
    function transferOwnership(uint256 _prpsIdx) public isOwner isPrpsExists(_prpsIdx) multiSig(_prpsIdx)
{

        // is right enum proposalType
        require(proposals[_prpsIdx].prpsType == ProposalType.transferOwner, "");
        address oldOwnerAddr = proposals[_prpsIdx].fromAddr;
        address newOwnerAddr = proposals[_prpsIdx].toAddr;
```

```
        require(oldOwnerAddr != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成
```

Gas 的消耗

```
        require(newOwnerAddr != address(0), ""); //SlowMist// 这类检查很好，避免操作失误导致合约
```

控制权丢失

```
        require(oldOwnerAddr != newOwnerAddr, "");
        for(uint256 i = 0; i < owners.length; i++) {
            if( owners[i] == oldOwnerAddr){
                owners[i] = newOwnerAddr;
                delete isOwnerMap[oldOwnerAddr];
                isOwnerMap[newOwnerAddr] = true;
            }
        }
        proposals[_prpsIdx].finalized = true;
        emit OwnershipTransferred(oldOwnerAddr, newOwnerAddr);
    }

}



/**
 * @title Pausable
 */
```

//SlowMist// 在出现重大交易异常时可以暂停所有交易，值得称赞的做法

```
contract Pausable is Ownable {

    event Pause();
    event Unpause();

    bool public paused = false;

    modifier whenNotPaused {
        require(!paused, "");
        _;
    }
    modifier whenPaused {
```

```
        require(paused, "");
        _;
    }

    // Pause contract
    function pause() public isOwner whenNotPaused returns (bool) {
        paused = true;
        emit Pause();
        return true;
    }

    // Unpause contract
    function unpause() public isOwner whenPaused returns (bool) {
        paused = false;
        emit Unpause();
        return true;
    }

}


/**
 * @title ERC20 interface
 */
contract ERC20 {
    event Transfer(address indexed _from, address indexed _to, uint256 _amount);
    // _from: _owner _to: _spender
    event Approval(address indexed _from, address indexed _to, uint256 _amount);
    function totalSupply() public view returns (uint256);
    function balanceOf(address _owner) public view returns (uint256);
    function transfer(address _to, uint256 _value) public returns (bool);
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool);
    function approve(address _spender, uint256 _value) public returns (bool);
    function allowance(address _owner, address _spender) public view returns (uint256);
}



/**
 * @title ERC20Token
 */
contract ERC20Token is ERC20 {
```

```
    using SafeMath for uint256;

    mapping(address => uint256) balances;
    mapping(address => mapping (address => uint256)) allowed;
    uint256 public totalToken;

    function totalSupply() public view returns (uint256) {
        return totalToken;
    }

    function balanceOf(address _owner) public view returns (uint256) {
        require(_owner != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas 的消耗
        return balances[_owner];
    }

    // Transfer token by internal
    function _transfer(address _from, address _to, uint256 _value) internal {
        require(_to != address(0), ""); //SlowMist// 这类检查很好，避免用户失误导致 Token 转丢
        require(balances[_from] >= _value, "");

        balances[_from] = balances[_from].sub(_value);
        balances[_to] = balances[_to].add(_value);
        emit Transfer(_from, _to, _value);
    }

    function transfer(address _to, uint256 _value) public returns (bool) {
        require(_to != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas 的消耗
        _transfer(msg.sender, _to, _value);
        return true; //SlowMist// 返回值符合 EIP20 规范
    }

    function transferFrom(address _from, address _to, uint256 _value) public returns (bool){
```

```
        require(_from != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas 的消耗

        require(_to != address(0), ""); //SlowMist// 这类检查很好，避免用户失误导致 Token 转丢
        require(_value > 0, "");
        require(balances[_from] >= _value, "");
        require(allowed[_from][msg.sender] >= _value, "");

        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; //SlowMist// 返回值符合 EIP20 规范
    }

    function approve(address _spender, uint256 _value) public returns (bool){
        require(_spender != address(0), ""); //SlowMist// 这类检查很好，避免用户失误导致授权错误
        require(_value > 0, "");
        allowed[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value);
        return true; //SlowMist// 返回值符合 EIP20 规范

    }

    function allowance(address _owner, address _spender) public view returns (uint256){
        require(_owner != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas 的消耗

        require(_spender != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas 的消耗

        return allowed[_owner][_spender];
```

```
    }

}


/**
 * @title LVECoin
 */
contract LVECoin is ERC20Token, Pausable {

    string public  constant name        = "LVECoin";
    string public  constant symbol      = "LVE";
    uint256 public constant decimals    = 18;
    // issue all token
    uint256 private initialToken        = 2000000000 * (10 ** decimals);

    // _to: _freezeAddr
    event Freeze(address indexed _to);
    // _to: _unfreezeAddr
    event Unfreeze(address indexed _to);
    event WithdrawalEther(address indexed _to, uint256 _amount);

    // freeze account mapping
    mapping(address => bool) public freezeAccountMap;
    // wallet Address
    address private walletAddr;
    // owner sign threshold
    uint256 private signThreshold       = 3;

    constructor(address[] _initOwners, address _walletAddr) public{
        require(_initOwners.length == signThreshold, "");

        require(_walletAddr != address(0), ""); //SlowMist// 这类检查很好，避免操作失误设置错了
```

walletAddr 地址

```
        // init owners
        requiredSignNum = _initOwners.length.div(2).add(1);
        owners = _initOwners;
        for(uint i = 0; i < _initOwners.length; i++) {
            isOwnerMap[_initOwners[i]] = true;
        }
```

```
        totalToken = initialToken;
        walletAddr = _walletAddr;
        balances[msg.sender] = totalToken;
        emit Transfer(0x0, msg.sender, totalToken);
    }


    // is freezeable account
    modifier freezeable(address _addr) {
        require(_addr != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas 的消耗
        require(!freezeAccountMap[_addr], "");
        _;
    }

    function transfer(address _to, uint256 _value) public whenNotPaused freezeable(msg.sender) returns (bool) {
        require(_to != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas 的消耗
        return super.transfer(_to, _value);
    }
    function transferFrom(address _from, address _to, uint256 _value) public whenNotPaused freezeable(msg.sender) returns (bool) {
        require(_from != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas 的消耗


        require(_to != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas 的消耗
        return super.transferFrom(_from, _to, _value);
    }
    function approve(address _spender, uint256 _value) public whenNotPaused freezeable(msg.sender) returns (bool) {
```

```
        require(_spender != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas
的消耗

        return super.approve(_spender, _value);
    }

    // freeze account
    function freezeAccount(uint256 _prpsIdx) public isOwner isPrpsExists(_prpsIdx) multiSig(_prpsIdx)
returns (bool) {

        // is right enum proposalType
        require(proposals[_prpsIdx].prpsType == ProposalType.freeze, "");
        address freezeAddr = proposals[_prpsIdx].toAddr;

        require(freezeAddr != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas
的消耗

        // proposals execute over
        proposals[_prpsIdx].finalized = true;
        freezeAccountMap[freezeAddr] = true;
        emit Freeze(freezeAddr);
        return true;
    }

    // unfreeze account
    function unfreezeAccount(uint256 _prpsIdx) public isOwner isPrpsExists(_prpsIdx) multiSig(_prpsIdx)
returns (bool) {

        // is right enum proposalType
        require(proposals[_prpsIdx].prpsType == ProposalType.unfreeze, "");
        address freezeAddr = proposals[_prpsIdx].toAddr;

        require(freezeAddr != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas
的消耗

        // proposals execute over
        proposals[_prpsIdx].finalized = true;
        freezeAccountMap[freezeAddr] = false;
        emit Unfreeze(freezeAddr);
        return true;
```

```
    }

    // if send ether then send ether to owner
    function() public payable {
        require(msg.value > 0, "");
        walletAddr.transfer(msg.value);
        emit WithdrawalEther(walletAddr, msg.value);
    }

}
```

FoundingTeam.sol

//SlowMist// 合约不存在溢出、条件竞争问题

```
pragma solidity ^0.4.24;


// *------------------------------------------------------------------------*
//       __ _    ________   __________  _____   __
//      / /| |  / / ____/  / ____/ __ \/  _/ | / /
//     / / | | / / __/    / /   / / / // //  |/ /
//    / /__| |/ / /___   / /___/ /_/ // // /|  /
//   /_____/___/_____/   \____/\____/___/_/ |_/
// *------------------------------------------------------------------------*



/**
 * @title SafeMath
 */
```

//SlowMist// 使用了 OpenZeppelin 的 SafeMath 安全模块，值得称赞的做法

```
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) {
        uint256 c = a / b;
        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 Ownable
 */
contract Ownable {

    address public owner;

    // _from: oldOwner _to: newOwner
    event OwnershipTransferred(address indexed _from, address indexed _to);

    constructor() public{
        owner = msg.sender;
    }

    // Modifier onlyOwner
    modifier onlyOwner {
        require(msg.sender == owner, "");
        _;
    }

    // Transfer owner
    function transferOwnership(address _newOwner) public onlyOwner {
```

```
        require(_newOwner != address(0), ""); //SlowMist// 这类检查很好，避免操作失误导致合约控制权丢失

        emit OwnershipTransferred(owner, _newOwner);
        owner = _newOwner;
    }
}


/**
 * @title Pausable
 */
contract Pausable is Ownable {

    event Pause();
    event Unpause();

    bool public paused = false;

    modifier whenNotPaused {
        require(!paused, "");
        _;
    }
    modifier whenPaused {
        require(paused, "");
        _;
    }

    // Pause contract
    function pause() public onlyOwner whenNotPaused returns (bool) {
        paused = true;
        emit Pause();
        return true;
    }

    // Unpause contract
    function unpause() public onlyOwner whenPaused returns (bool) {
        paused = false;
        emit Unpause();
        return true;
```

```
    }

}


interface LVECoin {
    function transfer(address _to, uint256 _value) external returns(bool);
    function balanceOf(address _owner) external view returns (uint256);
}


/**
 * @title FoundingTeam
 */
contract FoundingTeam is Pausable {

    using SafeMath for uint256;

    // token contract
    LVECoin private tokenContract;
    // token issue all amount
    uint256 private totalToken              = 2000000000 * (10 ** 18);
    // token supply amount
    uint256 public  tokenSupplyQuota        = totalToken.mul(50).div(1000);
    // lock end time
    uint256 public tokenLockEndTime;
    // alreary sold token
    uint256 public tokensSold               = 0;


    // Investor list
    struct Investor{
        uint256 endTime;        // token locked end time
        address addr;           // locked address
        bool isLocked;          // is lock address
        uint256 lockAmount;     // locked token amount
        uint256 investAmount;   // invest token amount
    }
    // investor mapping
    mapping(address => Investor) public investorMap;
    // freeze account mapping
    mapping(address => bool) public freezeAccountMap;
```

```
    // investor address list
    address[] public investorsList;

    // _to: _locker
    event Lock(address indexed _to, uint256 _amount, uint _endTime);
    // _to: _unlocker
    event UnLock(address indexed _to, uint256 _amount);
     // _to: _freezeAddr
    event Freeze(address indexed _to);
    // _to: _unfreezeAddr
    event Unfreeze(address indexed _to);
    event WithdrawalToken(address indexed _to, uint256 _amount);
    event WithdrawalEther(address indexed _to, uint256 _amount);

    constructor(address _tokenAddr, uint256 _tokenLockEndTime) public{

        require(_tokenAddr != address(0), ""); //SlowMist// 这类检查很好，避免操作失误设置错了
```

tokenContract 的地址

```
        require(_tokenLockEndTime > now, "");
        tokenLockEndTime = _tokenLockEndTime;
        tokenContract = LVECoin(_tokenAddr);
    }


    // is token on sale
    modifier isOnSale() {
        // 供給總 Token 額度 > 目前已售出 token 數量 => true, 販售中
        require(tokenSupplyQuota > tokensSold, "");
        _;
    }
    // is freezeable account
    modifier freezeable(address _addr) {

        require(_addr != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas 的
```

消耗

```
        require(!freezeAccountMap[_addr], "");
        _;
    }
```

```
    // get contract own token amount
    function getContractTokenBalance() public view returns(uint256 _rContractTokenAmount){
        return tokenContract.balanceOf(address(this));
    }


    // transfer token and lock
    function transferTokenAndLock(address _beneficiary, uint256 _amount) public onlyOwner isOnSale
freezeable(_beneficiary){
        require(_beneficiary != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成
Gas 的消耗

        // 目前已售出 token 数量
        tokensSold = tokensSold.add(_amount);
        // 判斷是否有超過總供給 Token 額度
        require(tokenSupplyQuota >= tokensSold, "");
        // add investor token locktime
        addlockAccount(_beneficiary, tokenLockEndTime, _amount);
    }


    // Locked warehouse function
    function addlockAccount(address _lockAddr, uint256 _endTime, uint256 _lockAmount) internal
returns(bool){
        require(_lockAddr != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas
的消耗

        require(_endTime >= now, "");
        require(_lockAmount > 0, "");
        if(investorMap[_lockAddr].addr != _lockAddr){
            investorsList.push(_lockAddr);
        }
        Investor memory investor;
        investor.endTime = _endTime;
        investor.addr = _lockAddr;
        investor.isLocked = true;
        investor.lockAmount = investorMap[_lockAddr].lockAmount.add(_lockAmount);
        investor.investAmount = investorMap[_lockAddr].investAmount.add(_lockAmount);
```

```
        investorMap[_lockAddr] = investor;

        emit Lock(_lockAddr, _lockAmount, _endTime);
        return true;
    }

    // freeze account
    function freezeAccount(address _freezeAddr) public onlyOwner returns (bool) {
        require(_freezeAddr != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成
```

Gas 的消耗

```
        freezeAccountMap[_freezeAddr] = true;
        emit Freeze(_freezeAddr);
        return true;
    }

    // unfreeze account
    function unfreezeAccount(address _freezeAddr) public onlyOwner returns (bool) {
        require(_freezeAddr != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成
```

Gas 的消耗

```
        freezeAccountMap[_freezeAddr] = false;
        emit Unfreeze(_freezeAddr);
        return true;
    }



    // get single investor invest information
    function getSingleInvestor(address _addr) public view returns(uint256 _rEndTime, address _rAddr, bool
_rIsLocked, uint256 _rLockAmount, uint256 _rInvestAmount){
        require(_addr != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成 Gas 的
```

消耗

```
        Investor memory investor = investorMap[_addr];
        return(investor.endTime, investor.addr, investor.isLocked, investor.lockAmount,
investor.investAmount);
    }
```

```
    // get investor count
    function getInvestorCount() public view returns(uint256 _rInvestorCount){
        return investorsList.length;
    }

    // get investor address
    function getInvestorAddr(uint256 _index) public view returns(address _rInvestorAddr){
        return investorsList[_index];
    }

    // after tokenLockEndTime owner Token
    function withdrawTokenToInvestorOwner(address _investorAddr) public onlyOwner returns(bool){

        require(_investorAddr != address(0), ""); //SlowMist// 这类检查很好，避免操作无用地址造成
```

Gas 的消耗

```
        require(now > tokenLockEndTime, "");
        Investor memory investor = investorMap[_investorAddr];
        if(investor.isLocked && now > investor.endTime && !freezeAccountMap[investor.addr]){
            require(tokenContract.transfer(investor.addr, investor.lockAmount), "");
            emit WithdrawalToken(investor.addr, investor.lockAmount);
            investor.endTime = 0;
            investor.isLocked = false;
            investor.lockAmount = 0;
            investorMap[investor.addr] = investor;
            emit UnLock(investor.addr, investor.lockAmount);
        }
        return true;
    }

    // after tokenLockEndTime batch Token
    function withdrawBatchTokenToInvestor() public onlyOwner returns(bool){
        require(now > tokenLockEndTime, "");
        // count investor nums
        uint256 investorCount = getInvestorCount();
        require(investorCount > 0, "");
        for (uint256 i = 0; i < investorCount; i++) {
            address investorAddr = investorsList[i];
            Investor memory investor = investorMap[investorAddr];
            if(investor.isLocked && now > investor.endTime && !freezeAccountMap[investor.addr]){

                require(tokenContract.transfer(investor.addr, investor.lockAmount), "");
```

```
                emit WithdrawalToken(investor.addr, investor.lockAmount);
                investor.endTime = 0;
                investor.isLocked = false;
                investor.lockAmount = 0;
                investorMap[investor.addr] = investor;
                emit UnLock(investor.addr, investor.lockAmount);
            }
        }
        return true;
    }

    // recycling Remain Token to wallet address
    function recyclingRemainToken() public onlyOwner whenNotPaused returns(bool){
        require(now > tokenLockEndTime, "");
        uint256 remainToken = tokenSupplyQuota.sub(tokensSold);
        require(remainToken > 0, "");
        require (tokenContract.transfer(msg.sender, remainToken), "");
        pause();
        return true;
    }

}
```

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