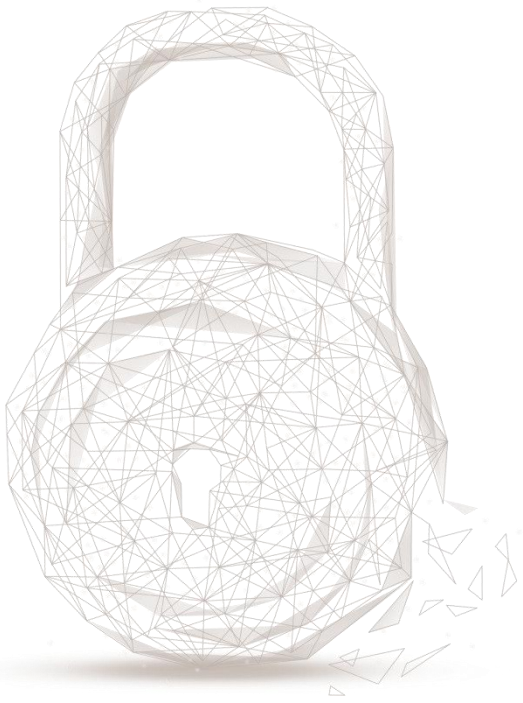
链安科技
Blockchain Security

Smart contract security audit report



Audit number：201808300930

Smart Contract name：

VoltToken

Smart Contract address：

Null

Smart Contract Link address：

https://github.com/VOLTTECHNOLOGY/SmartContract/blob/6a557a3f50f7ada480c90c17952f04b4cd6a16c2/VoltToken.sol

Start date of audit contract：2018.08.28

Completion date of audit contract：2018.08.30

Audit Conclusion：Pass (Excellent)

Audit team：Chengdu LianAn Technology Co. Ltd.

Audit type and results:

No.	Audit Type	Audit Subitems	Audit Results
1	Code For Programming Standardization Audit	ERC20 Token Standardization Audit	Pass
		Authority Declaration Rule Audit	Pass
		Gas Consumption Audit	Pass
		SafeMath Features Audit	Pass
		Fallback Usage Audit	Pass
2	Function Call Audit	Function Call Permission Audit	Pass
		Call Function Security Audit	Pass
		Delegatecall Function Security Audit	Pass
		Self-destruct Function Security Audit	Pass
3	Integer Overflow/Underflow Audit	-	Pass
4	Reentrancy Attack Audit	-	Pass
5	Incorrect reachable state	-	Pass

6	Execution-Ordering Dependency Audit	-	Pass
7	Timestamp Dependency Audit	-	Pass
8	tx.origin Audit	-	Pass
9	Token Vault Audit	-	Pass
10	Fake Deposit Audit	-	Pass
11	Events security Audit	-	Pass

Note: Audit results and suggestions in code comments

Disclaimer : This audit is only for the range of audit types given in the audit result table. Other unknown security vulnerabilities not listed in the table are beyond auditing responsibility. Chengdu LianAn Technology issues this report only based on the vulnerabilities that have already occurred or existed and takes corresponding responsibility in this regard. As for the new attacks or vulnerabilities that occur or exist in the future, Chengdu LianAn Technology cannot predict the security status of its smart contracts, thus taking no responsibility for them. The security audit analysis and other contents of this report are only based on the documents and materials provided by the contract provider to Chengdu LianAn Technology as of the issued time of this report, and no missing, falsified, deleted, or concealed documents and materials will be accepted. Contract provider should be aware that if the documents and materials provided are missing, falsified, deleted, concealed or inconsistent with the actual situation, Chengdu LianAn Technology disclaims any liability for the losses and negative effects caused by this reason.

Contract source code audit notes:

```
pragma solidity ^0.4.24;  // Chengdu LianAn // Recommend using a fixed version of compiler, and eliminate all compiler warnings

/**
 * @title SafeMath
 * @dev Math operations with safety checks that revert on error
 */
library SafeMath {
    /**
    * @dev Multiplies two numbers, reverts on overflow.
    */
    function mul(uint256 _a, uint256 _b) internal pure returns (uint256) {
        if (_a == 0) return 0;

        uint256 c = _a * _b;
        require(c / _a == _b);

        return c;
    }

    /**
    * @dev Integer division of two numbers truncating the quotient, reverts on division by zero.
    */
    function div(uint256 _a, uint256 _b) internal pure returns (uint256) {
        require(_b > 0);
        uint256 c = _a / _b;

        return c;
    }

    /**
    * @dev Subtracts two numbers, reverts on overflow (i.e. if subtrahend is greater than minuend).
    */
    function sub(uint256 _a, uint256 _b) internal pure returns (uint256) {
        require(_b <= _a);
        uint256 c = _a - _b;

        return c;
    }

    /**
    * @dev Adds two numbers, reverts on overflow.
    */
    function add(uint256 _a, uint256 _b) internal pure returns (uint256) {
```

```
        uint256 c = _a + _b;
        require(c >= _a);

        return c;
    }

    /**
    * @dev Divides two numbers and returns the remainder (unsigned integer modulo),
    * reverts when dividing by zero.
    */
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
        return a % b;
    }
}

contract BasicToken {
    string private token_name;
    string private token_symbol;
    uint8 private token_decimals;

    uint256 private total_supply;
    uint256 private remaining_supply;

    mapping (address => uint256) private balance_of;
    mapping (address => mapping(address => uint256)) private allowance_of;

    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed target, address indexed spender, uint256 value);

    constructor (
        string tokenName,
        string tokenSymbol,
        uint8 tokenDecimals,
        uint256 tokenSupply
    ) public {
        token_name = tokenName;
        token_symbol = tokenSymbol;
        token_decimals = tokenDecimals;
        total_supply = tokenSupply * (10 ** uint256(token_decimals));
        remaining_supply = total_supply;
    }

    function name() public view returns (string) {
        return token_name;
    }
```

```
    function symbol() public view returns (string) {
        return token_symbol;
    }

    function decimals() public view returns (uint8) {
        return token_decimals;
    }

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

    function remainingSupply() internal view returns (uint256) {
        return remaining_supply;
    }

    function balanceOf(
        address client_address
    ) public view returns (uint256) {
        return balance_of[client_address];
    }

    function setBalance(
        address client_address,
        uint256 value
    ) internal returns (bool) {
        if(client_address == address(0)) return false;
        balance_of[client_address] = value;
        return true;
    }

    function allowance(
        address target_address,
        address spender_address
    ) public view returns (uint256) {
        return allowance_of[target_address][spender_address];
    }
    // Chengdu LianAn // 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.
    // Chengdu LianAn // One possible solution to mitigate this  race condition is to first reduce the spender's allowance to 0 and set the desired value afterwards
    function approve(
        address spender_address,
        uint256 value
```

```
    ) public returns (bool) {
        require(value >= 0);
        require(msg.sender != address(0));
        require(spender_address != address(0));

        setApprove(msg.sender, spender_address, value);
        emit Approval(msg.sender, spender_address, value);
        return true;
    }

    function setApprove(
        address target_address,
        address spender_address,
        uint256 value
    ) internal returns (bool) {
        require(value >= 0);
        require(msg.sender != address(0));
        require(spender_address != address(0));

        allowance_of[target_address][spender_address] = value;
        return true;
    }

    function changeRemainingSupply(
        uint256 newRemainingSupply
    ) internal returns (bool) {
        remaining_supply = newRemainingSupply;
        return true;
    }
}

// Chengdu LianAn // owner management contract
contract VoltOwned {
    mapping (address => uint) private voltOwners;
    address[] private ownerList;

    mapping (address => uint256) private voltFreeze;

    modifier onlyOwner {
        require(voltOwners[msg.sender] == 99);
        _;
    }

    modifier noFreeze {
        require(now >= voltFreeze[msg.sender]);
        _;
```

```
    }

    constructor(address firstOwner) public {
        voltOwners[firstOwner] = 99;
        ownerList.push(firstOwner);
    }

    function isOwner(address who) internal view returns (bool) {
        if (voltOwners[who] == 99) {
            return true;
        } else {
            return false;
        }
    }

    function addOwner(address newVoltOwnerAddress) public onlyOwner noFreeze {
        require(newVoltOwnerAddress != address(0));
        require(voltOwners[newVoltOwnerAddress] != 99);
        voltOwners[newVoltOwnerAddress] = 99;
        ownerList.push(newVoltOwnerAddress);
    }

    function removeOwner(address removeVoltOwnerAddress) public onlyOwner noFreeze {
        require(removeVoltOwnerAddress != address(0));
        require(ownerList.length > 1);

        voltOwners[removeVoltOwnerAddress] = 0;
        for (uint256 i = 0; i != ownerList.length; i++) {
            if (removeVoltOwnerAddress == ownerList[i]) {
                delete ownerList[i];
                break;
            }
        }
    }

    function getOwners() view onlyOwner noFreeze returns (address[]) { return
ownerList; }

    function isFreeze(address who) internal view returns (bool) {
        if (now >= voltFreeze[who]) {
            return false;
        } else {
            return true;
        }
    }
```

```
    function setFreeze(
        address freezeAddress,
        uint256 timestamp
    ) public onlyOwner noFreeze returns (bool) {
        if (freezeAddress == address(0)) return false;
        voltFreeze[freezeAddress] = timestamp;
        return true;
    }

    function getFreezeTimestamp(address who) view onlyOwner noFreeze returns
(uint256) { return voltFreeze[who]; }
}

contract VoltToken is BasicToken, VoltOwned {
    using SafeMath for uint256;

    bool public mintStatus;

    event Deposit(address indexed from, address indexed to, uint256 value);
    event Mint(address indexed to, uint256 value);
    event Burn(address indexed target, uint256 value);

    constructor() public BasicToken (
        "VOLT", "ACDC", 18, 4000000000
    ) VoltOwned(
        msg.sender
    ) {
        mintStatus = true;
    }

    modifier canMint {
        require(mintStatus == true);
        _;
    }
    // Chengdu LianAn// Start minting and freeze the address that receives the
minted tokens
    function mint(
        address to,
        uint256 value,
        uint256 freezeTimestamp
    ) public onlyOwner noFreeze canMint {
        uint256 ts = totalSupply();
        uint256 rs = remainingSupply();
        require(ts >= rs);
        require(freezeTimestamp >= 0);
```

```
        superMint(to, value);
        setFreeze(to, freezeTimestamp); //our policy will be freeze all volt token.
but freezeTimestamp not actually used. usually 0.
    }
    // Chengdu LianAn // Start minting without freezing the address that receives
the minted tokens
    function superMint(address to, uint256 value) public onlyOwner noFreeze canMint
{
        uint256 rs = remainingSupply();
        require(rs >= value);

        uint256 currentBalance = balanceOf(to);
        setBalance(to, currentBalance.add(value));
        setRemainingSupply(rs.sub(value));
        emit Transfer(0x0, to, value);
        emit Mint(to, value);
    }
    // Chengdu LianAn // End minting
    function mintClose() public onlyOwner noFreeze returns (bool) {
        require(mintStatus == true);
        mintStatus = false;
        return true;
    }

    function transfer(
        address to,
        uint256 value
    ) public noFreeze returns (bool) {
        require(value > 0);
        require(msg.sender != address(0));
        require(to != address(0));

        require(balanceOf(msg.sender) >= value);
        require(balanceOf(to).add(value) >= balanceOf(to));

        voltTransfer(msg.sender, to, value);
        return true;
    }

    function transferFrom(
        address from,
        address to,
        uint256 value
    ) public noFreeze returns(bool) {
        require(value > 0);
        require(msg.sender != address(0));
```

```
        require(from != address(0));
        require(to != address(0));

        require(isFreeze(from) == false);
        require(allowance(from, msg.sender) >= value);
        require(balanceOf(from) >= value);
        require(balanceOf(to).add(value) >= balanceOf(to));

        voltTransfer(from, to, value);

        uint256 remaining = allowance(from, msg.sender).sub(value);
        setApprove(from, msg.sender, remaining);
        return true;
    }

    function voltTransfer(
        address from,
        address to,
        uint256 value
    ) private noFreeze returns (bool) {
        uint256 preBalance = balanceOf(from);
        setBalance(from, balanceOf(from).sub(value));
        setBalance(to, balanceOf(to).add(value));
        emit Transfer(from, to, value);
        assert(balanceOf(from).add(value) == preBalance);
        return true;
    }

    function setRemainingSupply(
        uint256 newRemainingSupply
    ) private onlyOwner noFreeze returns (bool) { return
changeRemainingSupply(newRemainingSupply); }
    // Chengdu LianAn // Acquire the number of the rest tokens that are allowed to
be used for minting
    function getRemainingSupply() view onlyOwner noFreeze returns (uint256) { return
remainingSupply(); }
}


// Chengdu LianAn // Recommend that the main contract inherits from 'Pausable'
module, so that owner can pause all transactions when there is a serious exception.
```

链安科技

Blockchain Security

Official Website

https://lianantech.com

E-mail

vaas@lianantech.com

Twitter

https://twitter.com/LianAnTech