



Smart Contract Security Audit Report



The SlowMist Security Team received the LBXC team's application for smart contract security audit of the LUXBIO CELL on December 16, 2019. The following are the details and results of this smart contract security audit:

Token name :

LBXC

File name and HASH(SHA256) :

LBXC.sol: 706a5fdebc76b59d8df4aef5222732320e9d0ad3e4aed5004a3293b883bf7fd9

The audit items and results :

(Other unknown security vulnerabilities are not included in the audit responsibility scope)

No.	Audit Items	Audit Subclass	Audit Subclass Result
1	Overflow Audit	-	Passed
2	Race Conditions Audit	-	Passed
3	Authority Control Audit	Permission vulnerability audit	Passed
		Excessive auditing authority	Some Risks
4	Safety Design Audit	Zeppelin module safe use	Passed
		Compiler version security	Passed
		Hard-coded address security	Passed
		Fallback function safe use	Passed
		Show coding security	Passed
		Function return value security	Passed
		Call function security	Passed
5	Denial of Service Audit	-	Passed
6	Gas Optimization Audit	-	Passed
7	Design Logic Audit	-	Passed
8	"False Deposit" vulnerability Audit	-	Passed
9	Malicious Event Log Audit	-	Passed
10	Scoping and Declarations Audit	-	Passed

11	Replay Attack Audit	ECDSA's Signature Replay Audit	Passed
12	Uninitialized Storage Pointers Audit	-	Passed
13	Arithmetic Accuracy Deviation Audit	-	Passed

Audit Result : Passed

Audit Number : 0X001912200001

Audit Date : December 20, 2019

Audit Team : SlowMist Security Team

(**Statement** : SlowMist only issues this report based on the fact that has occurred or existed before the report is issued, and bears the corresponding responsibility in this regard. For the facts occur or exist later after the report, SlowMist cannot judge the security status of its smart contract. SlowMist is not responsible for it. The security audit analysis and other contents of this report are based on the documents and materials provided by the information provider to SlowMist as of the date of this report (referred to as "the provided information"). SlowMist assumes that: there has been no information missing, tampered, deleted, or concealed. If the information provided has been missed, modified, deleted, concealed or reflected and is inconsistent with the actual situation, SlowMist will not bear any responsibility for the resulting loss and adverse effects. SlowMist will not bear any responsibility for the background or other circumstances of the project.)

Summary: This is a token contract that contain the tokenVault section. The total amount of contract tokens can be changed, superOwner can burn tokens of any user through the burn function. HiddenOwner can mint tokens through the mint function, but there is a cap. SafeMath security module is used, which is a commendable approach. The contract does not have the Overflow and the Race Conditions issue.

During the audit, we found the following problems:

- 1. Locker transfers tokens when timelock is true, only the first transfer amount is locked, other transfer amount is not locked**
- 2. User(not locker) will not be able to transfer when timeLock is true and `_timeLimit() > 5`**
- 3. The user's lockValue will not be changed after all tokens are unlocked until the owner rest the lockValues through setLockValue function**
- 4. The owner can set the lockValues for the user(not locker) through the setLockvalue function.**

The source code:

```
/**
 *Submitted for verification at Etherscan.io on 2019-01-31
 */

//SlowMist// The contract does not have the Overflow and the Race Conditions issue

pragma solidity >0.4.99 <0.6.0;

contract ERC20Basic {
    function totalSupply() public view returns (uint256);
    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
    );
}

library SafeERC20 {
    function safeTransfer(
        ERC20Basic _token,
        address _to,
        uint256 _value
    ) internal
    {
```

```
require(_token.transfer(_to, _value));
}

function safeTransferFrom(
    ERC20 _token,
    address _from,
    address _to,
    uint256 _value
) internal
{
    require(_token.transferFrom(_from, _to, _value));
}

function safeApprove(
    ERC20 _token,
    address _spender,
    uint256 _value
) internal
{
    require(_token.approve(_spender, _value));
}
}
```

//SlowMist// SafeMath security module is used, which is a commendable approach

```
library SafeMath {
    /**
     * @dev Multiplies two numbers, throws on overflow.
     */
    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
        // Gas optimization: this is cheaper than asserting 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
        if(a == 0) {
            return 0;
        }
        c = a * b;
        assert(c / a == b); //SlowMist// It is recommended to replace "assert" with "require" to
optimize Gas
        return c;
    }
}
```

```
/**
 * @dev Integer division of two numbers, truncating the quotient.
 */
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 a / b;
}
```

```
/**
 * @dev Subtracts two numbers, throws on overflow (i.e. if subtrahend is greater than minuend).
 */
function sub(uint256 a, uint256 b) internal pure returns (uint256) {

    assert(b <= a); //SlowMist// It is recommended to replace "assert" with "require" to
```

optimize Gas

```
    return a - b;
}

/**
 * @dev Adds two numbers, throws on overflow.
 */
function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
    c = a + b;

    assert(c >= a); //SlowMist// It is recommended to replace "assert" with "require" to
```

optimize Gas

```
    return c;
}
}
```

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

```
mapping(address => uint256) balances;
```

```
uint256 totalSupply_;
```

```
/**
```

```
 * @dev Total number of tokens in existence
```

```
 */
```

```
function totalSupply() public view returns (uint256) {
```

```
    return totalSupply_;
```

```
}
```

```
/**
```

```
 * @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)); //SlowMist// This kind of check is very good, avoiding user mistake
```

leading to the loss of token during transfer

```
    require(_value <= balances[msg.sender]);
```

```
    balances[msg.sender] = balances[msg.sender].sub(_value);
```

```
    balances[_to] = balances[_to].add(_value);
```

```
    emit Transfer(msg.sender, _to, _value);
```

```
    return true; //SlowMist// The return value conforms to the EIP20 specification
```

```
}
```

```
    /**
```

```
 * @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) {
```

```
    return balances[_owner];
```

```
}
```

```
}
```

```
/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * https://github.com/ethereum/EIPs/issues/20
 * 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)); //SlowMist// This kind of check is very good, avoiding user mistake

leading to the loss of token during transfer

        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; //SlowMist// The return value conforms to the EIP20 specification
    }

    /**
     * @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; //SlowMist// The return value conforms to the EIP20 specification
}

/**
 * @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,
    uint256 _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,
    uint256 _subtractedValue
) public returns (bool)
{
    uint256 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 MultiOwnable
*
* @dev LBXC 의 MultiOwnable 은 히든오너, 수퍼오너, 버너, 오너, 리클레이머를 설정한다. 권한을 여러명에게 부여할 수 있는
경우
* 리스트에 그 값을 넣어 불특정 다수가 확인 할 수 있게 한다.
*
* LBXC 의 MultiOwnable 可设置 HIDDENOWNER, SUPEROWNER, BURNER, OWNER 及 RECLAIMER.
* 其权限可同时赋予多人的情况, 在列表中放入该值后可确认其非特定的多人名单.
*
* MulitOwnable of LBXC sets HIDDENOWNER, SUPEROWNER, BURNER, OWNER, and RECLAIMER.
* If many can be authorized, the value is entered to the list so that it is accessible to unspecified many.
*
*/
contract MultiOwnable {
    uint8 constant MAX_BURN = 3;
    uint8 constant MAX_OWNER = 15;
    address payable public hiddenOwner;
    address payable public superOwner;
    address payable public reclaimer;

    address[MAX_BURN] public chkBurnerList;
    address[MAX_OWNER] public chkOwnerList;

    mapping(address => bool) public burners;
    mapping (address => bool) public owners;

    event AddedBurner(address indexed newBurner);
    event AddedOwner(address indexed newOwner);
    event DeletedOwner(address indexed toDeleteOwner);
    event DeletedBurner(address indexed toDeleteBurner);
    event ChangedReclaimer(address indexed newReclaimer);
    event ChangedSuperOwner(address indexed newSuperOwner);
    event ChangedHiddenOwner(address indexed newHiddenOwner);

    constructor() public {
        hiddenOwner = msg.sender;
        superOwner = msg.sender;
        reclaimer = msg.sender;
        owners[msg.sender] = true;
        chkOwnerList[0] = msg.sender;
    }
}

```

```
modifier onlySuperOwner() {
    require(superOwner == msg.sender);
    _;
}
modifier onlyReclaimer() {
    require(reclaimer == msg.sender);
    _;
}
modifier onlyHiddenOwner() {
    require(hiddenOwner == msg.sender);
    _;
}
modifier onlyOwner() {
    require(owners[msg.sender]);
    _;
}
modifier onlyBurner(){
    require(burners[msg.sender]);
    _;
}
```

```
function changeSuperOwnership(address payable newSuperOwner) public onlyHiddenOwner returns(bool) {
    require(newSuperOwner != address(0)); //SlowMist// This check is quite good in avoiding losing
```

control of the contract caused by user mistakes

```
    superOwner = newSuperOwner;

    emit ChangedSuperOwner(superOwner);

    return true;
}

function changeHiddenOwnership(address payable newHiddenOwner) public onlyHiddenOwner returns(bool) {
    require(newHiddenOwner != address(0));
    hiddenOwner = newHiddenOwner;

    emit ChangedHiddenOwner(hiddenOwner);

    return true;
}

function changeReclaimer(address payable newReclaimer) public onlySuperOwner returns(bool) {
```

```
require(newReclaimer != address(0));
reclaimer = newReclaimer;

emit ChangedReclaimer(reclaimer);

return true;
}

function addBurner(address burner, uint8 num) public onlySuperOwner returns (bool) {
    require(num < MAX_BURN);
    require(burner != address(0));
    require(chkBurnerList[num] == address(0));
    require(burners[burner] == false);

    burners[burner] = true;
    chkBurnerList[num] = burner;

    emit AddedBurner(burner);

    return true;
}

function deleteBurner(address burner, uint8 num) public onlySuperOwner returns (bool){
    require(num < MAX_BURN);
    require(burner != address(0));
    require(chkBurnerList[num] == burner);

    burners[burner] = false;

    chkBurnerList[num] = address(0);

    emit DeletedBurner(burner);

    return true;
}

function addOwner(address owner, uint8 num) public onlySuperOwner returns (bool) {
    require(num < MAX_OWNER);
    require(owner != address(0));
    require(chkOwnerList[num] == address(0));
    require(owners[owner] == false);

    owners[owner] = true;
```

```
        chkOwnerList[num] = owner;

        emit AddedOwner(owner);

        return true;
    }

    function deleteOwner(address owner, uint8 num) public onlySuperOwner returns (bool) {
        require(num < MAX_OWNER);
        require(owner != address(0));
        require(chkOwnerList[num] == owner);
        owners[owner] = false;
        chkOwnerList[num] = address(0);

        emit DeletedOwner(owner);

        return true;
    }
}

/**
 * @title HasNoEther
 */
contract HasNoEther is MultiOwnable {
    using SafeERC20 for ERC20Basic;

    event ReclaimToken(address _token);

    /**
     * @dev Constructor that rejects incoming Ether
     * The `payable` flag is added so we can access `msg.value` without compiler warning. If we
     * leave out payable, then Solidity will allow inheriting contracts to implement a payable
     * constructor. By doing it this way we prevent a payable constructor from working. Alternatively
     * we could use assembly to access msg.value.
     */
    constructor() public payable {
        require(msg.value == 0);
    }

    /**
     * @dev Disallows direct send by settings a default function without the `payable` flag.
     */
    function() external {
```

```
}

function reclaimToken(ERC20Basic _token) external onlyReclaimer returns(bool){

    uint256 balance = _token.balanceOf(address(this));

    _token.safeTransfer(superOwner, balance);

    emit ReclaimToken(address(_token));

    return true;
}

}

contract Blacklist is MultiOwnable {

    mapping(address => bool) blacklisted;

    event Blacklisted(address indexed blacklist);
    event Whitelisted(address indexed whitelist);

    modifier whenPermitted(address node) {
        require(!blacklisted[node]);
        _;
    }

    function isPermitted(address node) public view returns (bool) {
        return !blacklisted[node];
    }

    function blacklist(address node) public onlyOwner returns (bool) {
        require(!blacklisted[node]);
        blacklisted[node] = true;
        emit Blacklisted(node);

        return blacklisted[node];
    }

    function unblacklist(address node) public onlySuperOwner returns (bool) {
```

```
        require(blacklisted[node]);
        blacklisted[node] = false;
        emit Whitelisted(node);

        return blacklisted[node];
    }
}

contract Burnlist is Blacklist {
    mapping(address => bool) public isburnlist;

    event Burnlisted(address indexed burnlist, bool signal);

    modifier isBurnlisted(address who) {
        require(isburnlist[who]);
        _;
    }

    function addBurnlist(address node) public onlyOwner returns (bool) {
        require(!isburnlist[node]);

        isburnlist[node] = true;

        emit Burnlisted(node, true);

        return isburnlist[node];
    }

    function delBurnlist(address node) public onlyOwner returns (bool) {
        require(isburnlist[node]);

        isburnlist[node] = false;

        emit Burnlisted(node, false);

        return isburnlist[node];
    }
}

contract PausableToken is StandardToken, HasNoEther, Burnlist {
```



```
uint8 constant MAX_LOCKER = 10;
bool public paused = false;
bool public timelock = false;
uint256 public openingTime;
address[MAX_LOCKER] public chkLockerList;

mapping(address => bool) public lockerAddrs;
mapping(address => uint256) public lockValues;

event SetLockValues(address addr, uint256 value);
event OnTimeLock(address who);
event OffTimeLock(address who);
event Paused(address addr);
event Unpaused(address addr);
event AddLocker(address addr);
event DelLocker(address addr);
event OpenedTime();

constructor() public {
    openingTime = block.timestamp;
}

modifier whenNotPaused() {
    require(!paused || owners[msg.sender]);
    _;
}

function addLocker (address locker, uint8 num) public onlySuperOwner returns (bool) {
    require(num < MAX_LOCKER);
    require(locker != address(0));
    require(!lockerAddrs[locker]);
    require(chkLockerList[num] == address(0));

    chkLockerList[num] = locker;
    lockerAddrs[locker] = true;

    emit AddLocker(locker);

    return lockerAddrs[locker];
}

function delLocker (address locker, uint8 num) public onlySuperOwner returns (bool) {
```

```
require(num < MAX_LOCKER);
require(locker != address(0));
require(lockerAddr[locker]);
require(chkLockerList[num] == locker);

chkLockerList[num] = address(0);
lockerAddr[locker] = false;

emit DelLocker(locker);

return lockerAddr[locker];
}
```

//SlowMist// Suspending all transactions upon major abnormalities is a recommended

approach

```
function pause() public onlySuperOwner returns (bool) {
    require(!paused);

    paused = true;

    emit Paused(msg.sender);

    return paused;
}

function unpause() public onlySuperOwner returns (bool) {
    require(paused);

    paused = false;

    emit Unpaused(msg.sender);

    return paused;
}

function onTimeLock() public onlySuperOwner returns (bool) {
    require(!timelock);
    timelock = true;
    emit OnTimeLock(msg.sender);

    return timelock;
}
```

```
}
```

```
function offTimeLock() public onlySuperOwner returns (bool) {
```

```
    require(timelock);
```

```
    timelock = false;
```

```
    emit OffTimeLock(msg.sender);
```

```
    return timelock;
```

```
}
```

```
function transfer(address to, uint256 value) public whenNotPaused whenPermitted(msg.sender) returns (bool) {
```

```
    //时间锁定的情况
```

```
    //타임락인 경우
```

```
    //when it is timelock
```

```
    if(timelock) {
```

```
        //msg.sender 为 lockerAddrs 的情况，接收者将更新被锁定的额度状态。
```

```
        //msg.sender 가 lockerAddrs 인 경우, 받은 사용자의 락된 발란스 상태를 업데이트해준다.
```

```
        //when msg.sender is lockerAddrs, the recipient's locked balance is updated.
```

```
        if(lockerAddrs[msg.sender]) {
```

```
            //lockerAddrs 向 to 发送的情况，最初金额将成为 lockValues。
```

```
            //lockerAddrs 가 to 에게 보내는 경우, 최초의 금액이 lockValues 가 된다.
```

```
            //when lockerAddrs sends to to, the initial amount becomes lockValues.
```

```
        //SlowMist// Only the first transfer amount is locked, other transfer amount is
```

```
not locked
```

```
        //SlowMist// After confirming with the project party, The design intends that the
```

```
initial amount transferred by the locker is set as lockValue
```

```
        if(lockValues[to] == 0) {
```

```
            lockValues[to] = value;
```

```
            emit SetLockValues(to, value);
```

```
        }
```

```
    return super.transfer(to, value);
```

```
//发送者为非 lockerAddr 的情况,
//보내는 사람이 lockerAddr 가 아닌 경우
//when sender is not lockerAddr
} else {

//发送者为非 lockerAddr, 且存在 lockValues 的情况
//보내는 사람이 lockerAddr 가 아니며, lockValues 가 있는 경우
//when sender is not lockerAddr, and has lockValues
if(lockValues[msg.sender] > 0) {

uint256 _totalAmount = balances[msg.sender];

uint256 lockValue = lockValues[msg.sender].div(5);

//需大于总价值 value 的限额 (总锁定金额 - 已解锁金额) 。
//전체 값의 value 를 제한 금액이 (전체 락된 금액 - 제한이 풀린 금액)보다 커야한다.
//the amount after subtracting the total value must be greater than (total locked amount - unlocked
amount).

//SlowMist// If _timeLimit() > 5 will not be able to transfer

//SlowMist// After confirming with the project party, it is intended to
manually turn off timelock on the 5th month

require(_totalAmount.sub(value) >= lockValues[msg.sender].sub(lockValue * _timeLimit()));

return super.transfer(to, value);

//发送者为非 lockerAddr, 且不存在 lockValues 的情况
//보내는 사람이 lockerAddr 가 아니며, lockValues 가 없는 경우
//when sender is not lockerAddr, and has no lockValues
} else {
return super.transfer(to, value);
}

}

//非时间锁定的情况
//타임락이 아닌 경우
//when it not timelock
} else {
return super.transfer(to, value);
```

```
    }  
}  
  
function transferFrom(address from, address to, uint256 value) public  
whenNotPaused whenPermitted(from) whenPermitted(msg.sender) returns (bool) {  
    require(!lockerAddrs[from]);  
  
    if(timelock) {  
  
        //lockValues[from]大于0的情况  
        //lockValues[from]이 0 보다 큰 경우  
        //when lockValues[from] is greater than 0  
        if(lockValues[from] > 0) {  
  
            uint256 _totalAmount = balances[from];  
  
            uint256 lockValue = lockValues[from].div(5);  
  
            //SlowMist// If _timeLimit() > 5 will not be able to transfer  
  
            //SlowMist// After confirming with the project party, it is intended to manually  
  
            turn off timelock on the 5th month  
  
            require(_totalAmount.sub(value) >= lockValues[from].sub(lockValue * _timeLimit()));  
  
            return super.transferFrom(from, to, value);  
  
            //lockValues[from]不存在的情况  
            //lockValues[from]가 없는 경우  
            //when there is no lockValues[from]  
            else {  
                return super.transferFrom(from, to, value);  
            }  
  
        } else {  
            return super.transferFrom(from, to, value);  
        }  
    }  
}  
  
function _timeLimit() internal view returns (uint256) {  
    uint256 presentTime = block.timestamp;
```

```
uint256 timeValue = presentTime.sub(openingTime);
uint256 _result = timeValue.div(31 days);
_result = _result.add(1);

return _result;
}

function setOpeningTime() public onlyHiddenOwner returns(bool) {

    openingTime = block.timestamp;

    emit OpenedTime();

    return true;
}

function getLimitPeriod() external view returns (uint256) {
    uint256 presentTime = block.timestamp;
    uint256 timeValue = presentTime.sub(openingTime);
    uint256 result = timeValue.div(31 days);
    result = result.add(1);
    return result;
}
```

//SlowMist// The user's lockValue will not be changed after all tokens are unlocked until the owner rest the lockValues through setLockValue function

//SlowMist// After confirming with the project party, owner can set lockvalue of the user to 0 using setLockValue function

//SlowMist// The owner can set the lockValues for the user(not locker) through the setLockvalue function

```
function setLockValue(address to, uint256 value) public onlyOwner returns (bool) {
    lockValues[to] = value;

    emit SetLockValues(to, value);

    return true;
}
```

```
}  
}  
/**  
 * @title LBXC  
 *  
 */  
contract LBXC is PausableToken {  
  
    event Burn(address indexed burner, uint256 value);  
    event Mint(address indexed minter, uint256 value);  
  
    string public constant name = "LUXBIO CELL";  
    uint8 public constant decimals = 18;  
    string public constant symbol = "LBXC";  
    uint256 public constant INITIAL_SUPPLY = 1e10 * (10 ** uint256(decimals));  
  
    constructor() public {  
        totalSupply_ = INITIAL_SUPPLY;  
        balances[msg.sender] = INITIAL_SUPPLY;  
  
        emit Transfer(address(0), msg.sender, INITIAL_SUPPLY);  
    }  
  
    function destory() public onlyHiddenOwner returns (bool) {  
  
        selfdestruct(superOwner);  
  
        return true;  
    }  
    /**  
     * @dev LBXC 의 민트는 오직 히든오너만 실행 가능하며, 수퍼오너에게 귀속된다.  
     * 추가로 발행하려는 토큰과 기존 totalSupply_의 합이 최초 발행된 토큰의 양(INITIAL_SUPPLY)보다 클 수 없다.  
     *  
     * LBXC 的 MINT 只能由 HIDDENOWNER 进行执行, 其所有权归 SUPEROWNER 所有。  
     * 追加进行发行的数字货币与 totalSupply_ 的和不可大于最初发行的数字货币(INITIAL_SUPPLY)数量。  
     *  
     * Only the Hiddenowner can mint LBXC, and the minted is reverted to SUPEROWNER.  
     * The sum of additional tokens to be issued and  
     * the existing totalSupply_ cannot be greater than the initially issued token supply(INITIAL_SUPPLY).  
     */  
    function mint(uint256 _amount) public onlyHiddenOwner returns (bool) {
```

```
require(INITIAL_SUPPLY >= totalSupply._add(_amount));

totalSupply_ = totalSupply._add(_amount);

balances[superOwner] = balances[superOwner].add(_amount);

emit Mint(superOwner, _amount);

emit Transfer(address(0), superOwner, _amount);

return true;
}

/**
 * @dev LBXC의 토큰은 오직 버너만 실행 가능하며, Owner가 등록할 수 있는 Burnlist에 등록된 계정만 토큰을
 *      수 있다.
 *
 * LBXC의 BURN은 오직 BURNER에 의해 실행되며, OWNER는 Burnlist에 등록된 계정만 토큰을
 *      BURN할 수 있다.
 *
 * Only the BURNER can burn LBXC,
 * and only the tokens that can be burned are those on Burnlist account that Owner can register.
 */
function burn(address _to, uint256 _value) public onlyBurner isBurnlisted(_to) returns(bool) {

    _burn(_to, _value);

    return true;
}

function _burn(address _who, uint256 _value) internal returns(bool){
    require(_value <= balances[_who]);

    balances[_who] = balances[_who].sub(_value);
    totalSupply_ = totalSupply_.sub(_value);

    emit Burn(_who, _value);
    emit Transfer(_who, address(0), _value);

    return true;
}
}
```




Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

[@SlowMist_Team](https://twitter.com/SlowMist_Team)

WeChat Official Account

