



智能合约安全审计报告



慢雾安全团队于 2019-04-29 日，收到 ART 团队对 AKCToken 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：

AKC

合约文件：

```
├── contracts
│   ├── AKC.sol
│   ├── ApproveAndCallFallBack.sol
│   ├── Controlled.sol
│   ├── ERC223.sol
│   ├── ERC223ReceivingContract.sol
│   └── TokenController.sol
├── libs
│   ├── ds-auth
│   │   └── auth.sol
│   ├── ds-math
│   │   └── math.sol
│   ├── ds-note
│   │   └── note.sol
│   ├── ds-stop
│   │   └── stop.sol
│   ├── ds-token
│   │   ├── base.sol
│   │   └── token.sol
│   └── erc20
│       └── erc20.sol
```

项目地址：

<https://github.com/artwook/contracts/blob/a875c01f82fce88b1c5c3a7163c19c853a4cbde8/contracts/AKC.sol>

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：0X001905060001

审计日期：2019 年 05 月 06 日

审计团队：慢雾安全团队

(声明：慢雾仅就本报告出具前已经发生或存在的事实出具本报告，并就此承担相应责任。对于出具以后发生或存在的事实，慢雾无法判断其智能合约安全状况，亦不对此承担责任。本报告所作的审计分析及其他内容，仅基于信息提供者截至本报告出具时向慢雾提供的文件和资料（简称“已提供资料”）。慢雾假设：已提供资料不存在缺失、被篡改、删减或隐瞒的情形。如已提供资料信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符的，慢雾对由此而导致的损失和不利影响不承担任何责任。慢雾仅对该项目的安全情况进行约定内的安全审计并出具了本报告，慢雾不对该项目背景及其他情况进行负责。)

总结：此为代币(token)合约，不包含锁仓(tokenVault)部分，合约使用了 DSMath 安全模块，值得称

赞的做法，合约不存在溢出、条件竞争问题，代币可增发，用户可以燃烧自己的代币，综合评估合约无风险。

合约源代码如下：

libs/ds-auth/auth.sol

//SlowMist// 合约不存在溢出、条件竞争问题

//SlowMist// 使用了 DSMath 安全模块，值得称赞的做法

```
// This program is free software: you can redistribute it and/or modify  
// it under the terms of the GNU General Public License as published by  
// the Free Software Foundation, either version 3 of the License, or  
// (at your option) any later version.
```

```
// This program is distributed in the hope that it will be useful,  
// but WITHOUT ANY WARRANTY; without even the implied warranty of  
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
// GNU General Public License for more details.
```

```
// You should have received a copy of the GNU General Public License  
// along with this program. If not, see <http://www.gnu.org/licenses/>.
```

```
pragma solidity ^0.4.23;
```

```
contract DSAuthority {  
    function canCall(  
        address src, address dst, bytes4 sig  
    ) public view returns (bool);  
}
```

```
contract DSAuthEvents {  
    event LogSetAuthority (address indexed authority);  
    event LogSetOwner      (address indexed owner);  
}
```

```
contract DSAuth is DSAuthEvents {  
    DSAuthority public authority;  
    address      public owner;  
  
    constructor() public {  
        owner = msg.sender;  
    }  
}
```

```
        emit LogSetOwner(msg.sender);
    }

    function setOwner(address owner_)
        public
        auth
    {
        owner = owner_;
        emit LogSetOwner(owner);
    }

    function setAuthority(DSAuthority authority_)
        public
        auth
    {
        authority = authority_;
        emit LogSetAuthority(authority);
    }

    modifier auth {
        require(isAuthorized(msg.sender, msg.sig));
        _;
    }

    function isAuthorized(address src, bytes4 sig) internal view returns (bool) {
        if (src == address(this)) {
            return true;
        } else if (src == owner) {
            return true;
        } else if (authority == DSAuthority(0)) {
            return false;
        } else {
            return authority.canCall(src, this, sig);
        }
    }
}
```

libs/ds-note/note.sol

```
/// note.sol -- the `note` modifier, for logging calls as events

// This program is free software: you can redistribute it and/or modify
```

```
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
```

```
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
```

```
// You should have received a copy of the GNU General Public License
// along with this program. If not, see <http://www.gnu.org/licenses/>.
```

```
pragma solidity ^0.4.23;
```

```
contract DSNote {
    event LogNote(
        bytes4 indexed sig,
        address indexed guy,
        bytes32 indexed foo,
        bytes32 indexed bar,
        uint wad,
        bytes fax
    ) anonymous;

    modifier note {
        bytes32 foo;
        bytes32 bar;

        assembly {
            foo := calldataload(4)
            bar := calldataload(36)
        }

        emit LogNote(msg.sig, msg.sender, foo, bar, msg.value, msg.data);
    }
}
```

```
libs/ds-stop/stop.sol
```

```
/// stop.sol -- mixin for enable/disable functionality
```

```
// Copyright (C) 2017 DappHub, LLC

// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.

// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this program. If not, see <http://www.gnu.org/licenses/>.
```

```
pragma solidity ^0.4.23;
```

//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

```
import "../ds-auth/auth.sol";
import "../ds-note/note.sol";

contract DSStop is DSNote, DSAuth {

    bool public stopped;

    modifier stoppable {
        require(!stopped);
        _;
    }

    function stop() public auth note {
        stopped = true;
    }

    function start() public auth note {
        stopped = false;
    }

}
```

```
libs/erc20/erc20.sol
```

```
/// erc20.sol -- API for the ERC20 token standard
```

```
// See <https://github.com/ethereum/EIPs/issues/20>.

// This file likely does not meet the threshold of originality
// required for copyright to apply. As a result, this is free and
// unencumbered software belonging to the public domain.

pragma solidity ^0.4.8;

contract ERC20Events {
    event Approval(address indexed src, address indexed guy, uint wad);
    event Transfer(address indexed src, address indexed dst, uint wad);
}

contract ERC20 is ERC20Events {
    function totalSupply() public view returns (uint);
    function balanceOf(address guy) public view returns (uint);
    function allowance(address src, address guy) public view returns (uint);

    function approve(address guy, uint wad) public returns (bool);
    function transfer(address dst, uint wad) public returns (bool);
    function transferFrom(
        address src, address dst, uint wad
    ) public returns (bool);
}
```

libs/ds-math/math.sol

```
/// math.sol -- mixin for inline numerical wizardry

// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.

// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this program. If not, see <http://www.gnu.org/licenses/>.
```


//SlowMist// 使用了 DSMath 安全模块，值得称赞的做法

```
pragma solidity ^0.4.13;

contract DSMath {
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x);
    }
    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x);
    }
    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x);
    }

    function min(uint x, uint y) internal pure returns (uint z) {
        return x <= y ? x : y;
    }
    function max(uint x, uint y) internal pure returns (uint z) {
        return x >= y ? x : y;
    }
    function imin(int x, int y) internal pure returns (int z) {
        return x <= y ? x : y;
    }
    function imax(int x, int y) internal pure returns (int z) {
        return x >= y ? x : y;
    }

    uint constant WAD = 10 ** 18;
    uint constant RAY = 10 ** 27;

    function wmul(uint x, uint y) internal pure returns (uint z) {
        z = add(mul(x, y), WAD / 2) / WAD;
    }
    function rmul(uint x, uint y) internal pure returns (uint z) {
        z = add(mul(x, y), RAY / 2) / RAY;
    }
    function wdiv(uint x, uint y) internal pure returns (uint z) {
        z = add(mul(x, WAD), y / 2) / y;
    }
    function rdiv(uint x, uint y) internal pure returns (uint z) {
        z = add(mul(x, RAY), y / 2) / y;
    }
}
```

```
}

// This famous algorithm is called "exponentiation by squaring"
// and calculates  $x^n$  with  $x$  as fixed-point and  $n$  as regular unsigned.
//
// It's  $O(\log n)$ , instead of  $O(n)$  for naive repeated multiplication.
//
// These facts are why it works:
//
// If  $n$  is even, then  $x^n = (x^2)^{(n/2)}$ .
// If  $n$  is odd, then  $x^n = x * x^{(n-1)}$ ,
// and applying the equation for even  $x$  gives
//  $x^n = x * (x^2)^{((n-1) / 2)}$ .
//
// Also, EVM division is flooring and
//  $\text{floor}[(n-1) / 2] = \text{floor}[n / 2]$ .
//
function rpow(uint x, uint n) internal pure returns (uint z) {
    z = n % 2 != 0 ? x : RAY;

    for (n /= 2; n != 0; n /= 2) {
        x = rmul(x, x);

        if (n % 2 != 0) {
            z = rmul(z, x);
        }
    }
}
```

libs/ds-token/base.sol

```
/// base.sol -- basic ERC20 implementation

// Copyright (C) 2015, 2016, 2017 DappHub, LLC

// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.

// This program is distributed in the hope that it will be useful,
```

```
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this program. If not, see <http://www.gnu.org/licenses/>.

pragma solidity ^0.4.23;

import "../erc20/erc20.sol";
import "../ds-math/math.sol";

contract DSTokenBase is ERC20, DSMath {
    uint256 _supply;
    mapping (address => uint256) _balances;
    mapping (address => mapping (address => uint256)) _approvals;

    constructor(uint supply) public {
        _balances[msg.sender] = supply;
        _supply = supply;
    }

    function totalSupply() public view returns (uint) {
        return _supply;
    }

    function balanceOf(address src) public view returns (uint) {
        return _balances[src];
    }

    function allowance(address src, address guy) public view returns (uint) {
        return _approvals[src][guy];
    }

    function transfer(address dst, uint wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad); //SlowMist// 返回值符合 EIP20 规范
    }

    function transferFrom(address src, address dst, uint wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
```

```
        _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
    }

    _balances[src] = sub(_balances[src], wad);
    _balances[dst] = add(_balances[dst], wad);

    emit Transfer(src, dst, wad);

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

function approve(address guy, uint wad) public returns (bool) {
    _approvals[msg.sender][guy] = wad;

    emit Approval(msg.sender, guy, wad);

    return true; //SlowMist// 返回值符合 EIP20 规范
}
}
```

libs/ds-token/token.sol

```
/// token.sol -- ERC20 implementation with minting and burning

// Copyright (C) 2015, 2016, 2017 DappHub, LLC

// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.

// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this program. If not, see <http://www.gnu.org/licenses/>.

pragma solidity ^0.4.23;
```

```
import "../ds-stop/stop.sol";

import "../base.sol";

contract DSToken is DSTokenBase(0), DSStop {

    bytes32 public symbol;
    uint256 public decimals = 18; // standard token precision. override to customize

    constructor(bytes32 symbol_) public {
        symbol = symbol_;
    }

    event Mint(address indexed guy, uint wad);
    event Burn(address indexed guy, uint wad);

    /* 这个方法会将所有余额授权给相应的账号 此处删除比较危险的功能 */
    /* function approve(address guy) public stoppable returns (bool) {
            return super.approve(guy, uint(-1));
    */

    function approve(address guy, uint wad) public stoppable returns (bool) {
        return super.approve(guy, wad);
    }

    function transferFrom(address src, address dst, uint wad)
        public
        stoppable
        returns (bool)
    {
        if (src != msg.sender && _approvals[src][msg.sender] != uint(-1)) {
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad);
        }

        _balances[src] = sub(_balances[src], wad);
        _balances[dst] = add(_balances[dst], wad);

        emit Transfer(src, dst, wad);

        return true;
    }
}
```

```
function push(address dst, uint wad) public {
    transferFrom(msg.sender, dst, wad);
}
function pull(address src, uint wad) public {
    transferFrom(src, msg.sender, wad);
}
function move(address src, address dst, uint wad) public {
    transferFrom(src, dst, wad);
}
```

```
function mint(uint wad) public {
    mint(msg.sender, wad);
}
function burn(uint wad) public {
    burn(msg.sender, wad);
}
```

//SlowMist// 铸币功能

```
function mint(address guy, uint wad) public auth stoppable {
    _balances[guy] = add(_balances[guy], wad);
    _supply = add(_supply, wad);
    emit Mint(guy, wad);
}
```

//SlowMist// 燃烧功能，如果代理商作恶，存在被恶意燃烧的可能

```
function burn(address guy, uint wad) public auth stoppable {
    if (guy != msg.sender && _approvals[guy][msg.sender] != uint(-1)) {
        _approvals[guy][msg.sender] = sub(_approvals[guy][msg.sender], wad);
    }

    _balances[guy] = sub(_balances[guy], wad);
    _supply = sub(_supply, wad);
    emit Burn(guy, wad);
}
```

// Optional token name

```
bytes32 public name = "";
```

```
function setName(bytes32 name_) public auth {
    name = name_;
}
```

```
}
```

contracts/ERC223ReceivingContract.sol

```
pragma solidity ^0.4.13;

/*
 * Contract that is working with ERC223 tokens
 * https://github.com/ethereum/EIPs/issues/223
 */

/// @title ERC223ReceivingContract - Standard contract implementation for compatibility with ERC223 tokens.
contract ERC223ReceivingContract {

    /// @dev Function that is called when a user or another contract wants to transfer funds.
    /// @param _from Transaction initiator, analogue of msg.sender
    /// @param _value Number of tokens to transfer.
    /// @param _data Data containig a function signature and/or parameters
    function tokenFallback(address _from, uint256 _value, bytes _data) public;

    /// @dev For ERC20 backward compatibility, same with above tokenFallback but without data.
    /// The function execution could fail, but do not influence the token transfer.
    /// @param _from Transaction initiator, analogue of msg.sender
    /// @param _value Number of tokens to transfer.
    // function tokenFallback(address _from, uint256 _value) public;
}
```

contracts/TokenController.sol

```
pragma solidity ^0.4.13;

/// @dev The token controller contract must implement these functions
contract TokenController {

    /// @notice Called when `_owner` sends ether to the MiniMe Token contract
    /// @param _owner The address that sent the ether to create tokens
    /// @return True if the ether is accepted, false if it throws
    function proxyPayment(address _owner) payable public returns (bool);

    /// @notice Notifies the controller about a token transfer allowing the
    /// controller to react if desired
}
```

```
/// @param _from The origin of the transfer
/// @param _to The destination of the transfer
/// @param _amount The amount of the transfer
/// @return False if the controller does not authorize the transfer
function onTransfer(address _from, address _to, uint _amount) public returns (bool);

/// @notice Notifies the controller about an approval allowing the
/// controller to react if desired
/// @param _owner The address that calls `approve()`
/// @param _spender The spender in the `approve()` call
/// @param _amount The amount in the `approve()` call
/// @return False if the controller does not authorize the approval
function onApprove(address _owner, address _spender, uint _amount) public returns (bool);
}
```

contracts/Controlled.sol

```
pragma solidity ^0.4.13;

contract Controlled {
    /// @notice The address of the controller is the only address that can call
    /// a function with this modifier
    modifier onlyController { if (msg.sender != controller) revert(); _; }

    address public controller;

    constructor() { controller = msg.sender;}

    /// @notice Changes the controller of the contract
    /// @param _newController The new controller of the contract
    function changeController(address _newController) onlyController {
        controller = _newController;
    }
}
```

contracts/ApproveAndCallFallback.sol

```
pragma solidity ^0.4.13;

contract ApproveAndCallFallback {
    function receiveApproval(address from, uint256 _amount, address _token, bytes _data);
}
```


contracts/ERC223.sol

```
pragma solidity ^0.4.13;

contract ERC223 {
    function transfer(address to, uint amount, bytes data) public returns (bool ok);

    function transferFrom(address from, address to, uint256 amount, bytes data) public returns (bool ok);

    function transfer(address to, uint amount, bytes data, string custom_fallback) public returns (bool ok);

    function transferFrom(address from, address to, uint256 amount, bytes data, string custom_fallback)
    public returns (bool ok);

    event ERC223Transfer(address indexed from, address indexed to, uint amount, bytes data);

    event ReceivingContractTokenFallbackFailed(address indexed from, address indexed to, uint amount);
}
```

contracts/AKC.sol

```
pragma solidity ^0.4.25;

import "../libs/ds-token/token.sol";
import "../ERC223ReceivingContract.sol";
import "../TokenController.sol";
import "../Controlled.sol";
import "../ApproveAndCallFallBack.sol";
import "../ERC223.sol";

contract AKC is DSToken("AKC"), ERC223, Controlled {

    uint256 public cap = 2e26;

    constructor() {
        setName("ARTWOOK Coin");
    }

    /// @notice Send `_amount` tokens to `_to` from `_from` on the condition it
    /// is approved by `_from`
    /// @param _from The address holding the tokens being transferred
    /// @param _to The address of the recipient
```

```
/// @param _amount The amount of tokens to be transferred
/// @return True if the transfer was successful
function transferFrom(address _from, address _to, uint256 _amount
) public returns (bool success) {
    // Alerts the token controller of the transfer
    if (isContract(controller)) {
        if (!TokenController(controller).onTransfer(_from, _to, _amount))
            revert();
    }

    success = super.transferFrom(_from, _to, _amount);

    if (success && isContract(_to))
    {
        // ERC20 backward compatiability
        if(!_to.call(bytes4(keccak256("tokenFallback(address,uint256)")), _from, _amount)) {
            // do nothing when error in call in case that the _to contract is not inherited from
            ERC223ReceivingContract
                // revert();
                // bytes memory empty;

            emit ReceivingContractTokenFallbackFailed(_from, _to, _amount);

            // Even the fallback failed if there is such one, the transfer will not be revert since
            "revert()" is not called.
        }
    }
}

/*
 * ERC 223
 * Added support for the ERC 223 "tokenFallback" method in a "transfer" function with a payload.
 */
function transferFrom(address _from, address _to, uint256 _amount, bytes _data)
public
returns (bool success)
{
    // Alerts the token controller of the transfer
    if (isContract(controller)) {
        if (!TokenController(controller).onTransfer(_from, _to, _amount))
            revert();
    }
}
```

```
require(super.transferFrom(_from, _to, _amount));

if (isContract(_to)) {
    ERC223ReceivingContract receiver = ERC223ReceivingContract(_to);
    receiver.tokenFallback(_from, _amount, _data);
}

emit ERC223Transfer(_from, _to, _amount, _data);

return true;
}

/*
 * ERC 223
 * Added support for the ERC 223 "tokenFallback" method in a "transfer" function with a payload.
 * https://github.com/ethereum/EIPs/issues/223
 * function transfer(address _to, uint256 _value, bytes _data) public returns (bool success);
 */
/// @notice Send `_value` tokens to `_to` from `msg.sender` and trigger
/// tokenFallback if sender is a contract.
/// @dev Function that is called when a user or another contract wants to transfer funds.
/// @param _to Address of token receiver.
/// @param _amount Number of tokens to transfer.
/// @param _data Data to be sent to tokenFallback
/// @return Returns success of function call.
function transfer(
    address _to,
    uint256 _amount,
    bytes _data)
    public
    returns (bool success)
{
    return transferFrom(msg.sender, _to, _amount, _data);
}

/*
 * ERC 223
 * Added support for the ERC 223 "tokenFallback" method in a "transfer" function with a payload.
 */
function transferFrom(address _from, address _to, uint256 _amount, bytes _data, string _custom_fallback)
    public
```

```
returns (bool success)
{
    // Alerts the token controller of the transfer
    if (isContract(controller)) {
        if (!TokenController(controller).onTransfer(_from, _to, _amount))
            revert();
    }

    require(super.transferFrom(_from, _to, _amount));

    if (isContract(_to)) {
        /* 修复 ERC233 与 ds-auth 合用时 产生的安全漏洞 */

        if(_to == address(this)) revert(); //SlowMist// 这个检查很好，修复了已知的安全漏洞

        ERC223ReceivingContract receiver = ERC223ReceivingContract(_to);
        receiver.call.value(0)(bytes4(keccak256(_custom_fallback)), _from, _amount, _data);
    }

    emit ERC223Transfer(_from, _to, _amount, _data);

    return true;
}

/*
 * ERC 223
 * Added support for the ERC 223 "tokenFallback" method in a "transfer" function with a payload.
 */
function transfer(
    address _to,
    uint _amount,
    bytes _data,
    string _custom_fallback)
    public
    returns (bool success)
{
    return transferFrom(msg.sender, _to, _amount, _data, _custom_fallback);
}

/// @notice `msg.sender` approves `_spender` to spend `_amount` tokens on
/// its behalf. This is a modified version of the ERC20 approve function
/// to be a little bit safer
/// @param _spender The address of the account able to transfer the tokens
```

```
/// @param _amount The amount of tokens to be approved for transfer
/// @return True if the approval was successful
function approve(address _spender, uint256 _amount) returns (bool success) {
    // Alerts the token controller of the approve function call
    if (isContract(controller)) {
        if (!TokenController(controller).onApprove(msg.sender, _spender, _amount))
            revert();
    }

    return super.approve(_spender, _amount);
}

function mint(address _guy, uint _wad) auth stoppable {
    require(add(_supply, _wad) <= cap);

    super.mint(_guy, _wad);

    emit Transfer(0, _guy, _wad);
}

function burn(address _guy, uint _wad) auth stoppable {
    super.burn(_guy, _wad);

    emit Transfer(_guy, 0, _wad);
}

/// @notice `msg.sender` approves `_spender` to send `_amount` tokens on
/// its behalf, and then a function is triggered in the contract that is
/// being approved, `_spender`. This allows users to use their tokens to
/// interact with contracts in one function call instead of two
/// @param _spender The address of the contract able to transfer the tokens
/// @param _amount The amount of tokens to be approved for transfer
/// @return True if the function call was successful
function approveAndCall(address _spender, uint256 _amount, bytes _extraData
) returns (bool success) {
    if (!approve(_spender, _amount)) revert();

    ApproveAndCallFallback(_spender).receiveApproval(
        msg.sender,
        _amount,
        this,
        _extraData
    );
}
```

```
        return true;
    }

    /// @dev Internal function to determine if an address is a contract
    /// @param _addr The address being queried
    /// @return True if `_addr` is a contract
    function isContract(address _addr) constant internal returns(bool) {
        uint size;
        if (_addr == 0) return false;
        assembly {
            size := extcodesize(_addr)
        }
        return size>0;
    }

    /// @notice The fallback function: If the contract's controller has not been
    /// set to 0, then the `proxyPayment` method is called which relays the
    /// ether and creates tokens as described in the token controller contract
    function () payable {
        if (isContract(controller)) {
            if (! TokenController(controller).proxyPayment.value(msg.value)(msg.sender))
                revert();
        } else {
            revert();
        }
    }

    //////////
    // Safety Methods
    //////////

    /// @notice This method can be used by the controller to extract mistakenly
    /// sent tokens to this contract.
    /// @param _token The address of the token contract that you want to recover
    /// set to 0 in case you want to extract ether.
    function claimTokens(address _token) onlyController {
        if (_token == 0x0) {
            controller.transfer(this.balance);
            return;
        }
    }
```

```
ERC20 token = ERC20(_token);
uint balance = token.balanceOf(this);
/* 避免外部调用返回 false 代码仍然继续执行 此处加上 require 判断 */
require(token.transfer(controller, balance));
emit ClaimedTokens(_token, controller, balance);
}

////////////////
// Events
////////////////

event ClaimedTokens(address indexed _token, address indexed _controller, uint _amount);
}
```



官方网址

www.slowmist.com

电子邮箱

team@slowmist.com

微信公众号

