慢雾科技
slow mist

智能合约安全审计报告

慢雾安全团队于 2018-05-28 日，收到 FACTS 团队对 FACTS 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：

FACTS

合约地址：

0x09cb097356fd053f8544abfa2c8a9d4fb2200d62

链接地址：

https://etherscan.io/address/0x09cb097356fd053f8544abfa2c8a9d4fb2200d62#code

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	-	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
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

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：0X001806030001

审计日期：2018 年 06 月 03 日

审计团队：慢雾安全团队

合约源代码如下：

```
pragma solidity ^0.4.21;
```

//SlowMist// 合约不存在溢出、条件竞争问题

//SlowMist// 使用了大量 OpenZeppelin 的 SafeMath 及 ERC20 标准模块，值得称赞的做法

```
/**
 * Math operations with safety checks
 */
library SafeMath {
  function mul(uint a, uint b) internal pure returns (uint) {
    uint c = a * b;
    assert(a == 0 || c / a == b);
    return c;
  }

  function div(uint a, uint b) internal pure returns (uint) {
    // assert(b > 0); // Solidity automatically throws when dividing by 0
    uint c = a / b;
    assert(a == b * c + a % b); // There is no case in which this doesn't hold
    return c;
  }

  function sub(uint a, uint b) internal pure returns (uint) {
    assert(b <= a);
    return a - b;
  }

  function add(uint a, uint b) internal pure returns (uint) {
    uint c = a + b;
    assert(c >= a);
    return c;
  }

  function max64(uint64 a, uint64 b) internal pure returns (uint64) {
    return a >= b ? a : b;
  }

  function min64(uint64 a, uint64 b) internal pure returns (uint64) {
    return a < b ? a : b;
  }

  function max256(uint256 a, uint256 b) internal pure returns (uint256) {
    return a >= b ? a : b;
  }

  function min256(uint256 a, uint256 b) internal pure returns (uint256) {
    return a < b ? a : b;
  }

}


/**
 * @title ERC20Basic
 * @dev Simpler version of ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 */
contract ERC20Basic {
  uint public totalSupply;
  function balanceOf(address who) constant public returns (uint);
  function transfer(address to, uint value) public;
  event Transfer(address indexed from, address indexed to, uint value);
}


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

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

//SlowMist// 短地址攻击防护，值得称赞的做法

```
  /**
   * @dev Fix for the ERC20 short address attack.
   */
  modifier onlyPayloadSize(uint size) {
     assert(msg.data.length >= size + 4);
     _;
  }

  /**
  * @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, uint _value) onlyPayloadSize(2 * 32)  public {
    balances[msg.sender] = balances[msg.sender].sub(_value);
    balances[_to] = balances[_to].add(_value);
    emit Transfer(msg.sender, _to, _value);
  }

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

}



/**
 * @title ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 */
contract ERC20 is ERC20Basic {
  function allowance(address owner, address spender) constant  public returns (uint);
  function transferFrom(address from, address to, uint value)  public;
  function approve(address spender, uint value)  public;
  event Approval(address indexed owner, address indexed spender, uint value);
}
```

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

  mapping (address => mapping (address => uint)) 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 uint the amout of tokens to be transfered
   */
  function transferFrom(address _from, address _to, uint _value) onlyPayloadSize(3 * 32)  public {
    uint _allowance;
    _allowance = allowed[_from][msg.sender];

    require(_allowance >= _value);

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

  /**
   * @dev Aprove the passed address to spend the specified amount of tokens on beahlf of msg.sender.
   * @param _spender The address which will spend the funds.
   * @param _value The amount of tokens to be spent.
   */
  function approve(address _spender, uint _value)  public {

    // To change the approve amount you first have to reduce the addresses`
    //  allowance to zero by calling `approve(_spender, 0)` if it is not
    //  already 0 to mitigate the race condition described here:
```

```
    //  https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
    require((_value == 0) || (allowed[msg.sender][_spender] == 0));

    allowed[msg.sender][_spender] = _value;
    emit Approval(msg.sender, _spender, _value);
  }

  /**
   * @dev Function to check the amount of tokens than 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 uint specifing the amount of tokens still avaible for the spender.
   */
  function allowance(address _owner, address _spender) constant public returns (uint remaining) {
    return allowed[_owner][_spender];
  }

}


/**
 * @title Ownable
 * @dev The Ownable contract has an owner address, and provides basic authorization control
 * functions, this simplifies the implementation of "user permissions".
 */
contract Ownable {
  address public owner;


  /**
   * @dev The Ownable constructor sets the original `owner` of the contract to the sender
   * account.
   */
  constructor()  public {
    owner = msg.sender;
  }


  /**
   * @dev Throws if called by any account other than the owner.
   */
  modifier onlyOwner() {
    require(msg.sender == owner);
    _;
  }


  /**
   * @dev Allows the current owner to transfer control of the contract to a newOwner.
   * @param newOwner The address to transfer ownership to.
   */
  function transferOwnership(address newOwner) onlyOwner  public {
    if (newOwner != address(0)) { //SlowMist// 这个检查很好，避免操作失误失去合约控制权
      owner = newOwner;
    }
  }

}



/**
 * @title Mintable token
 * @dev Simple ERC20 Token example, with mintable token creation
 * @dev Issue: * https://github.com/OpenZeppelin/zeppelin-solidity/issues/120
 *              Based            on           code           by            TokenMarketNet:
https://github.com/TokenMarketNet/ico/blob/master/contracts/MintableToken.sol
 */

contract MintableToken is StandardToken, Ownable {
  event Mint(address indexed to, uint value);
  event MintFinished();

  bool public mintingFinished = false;
  uint public totalSupply = 0;

  modifier canMint() {
    require(!mintingFinished);
    _;
  }

  /**
   * @dev Function to mint tokens
   * @param _to The address that will recieve the minted tokens.
   * @param _amount The amount of tokens to mint.
   * @return A boolean that indicates if the operation was successful.
   */
  function mint(address _to, uint _amount) onlyOwner canMint  public returns (bool) {
    totalSupply = totalSupply.add(_amount);
    balances[_to] = balances[_to].add(_amount);
    emit Mint(_to, _amount);
    return true;
  }

  /**
   * @dev Function to stop minting new tokens.
   * @return True if the operation was successful.
   */
  function finishMinting() onlyOwner  public returns (bool) {
    mintingFinished = true;
    emit MintFinished();
    return true;
  }
}
```

//SlowMist// 当出现重大异常时可以暂停所有交易，值得称赞的做法

```
/**
 * @title Pausable
 * @dev Base contract which allows children to implement an emergency stop mechanism.
 */
contract Pausable is Ownable {
  event Pause();
  event Unpause();

  bool public paused = false;


  /**
   * @dev modifier to allow actions only when the contract IS paused
   */
  modifier whenNotPaused() {
    // if (paused) throw;
    require(!paused);
    _;
  }

  /**
   * @dev modifier to allow actions only when the contract IS NOT paused
   */
  modifier whenPaused {
    require(paused);
    _;
  }

  /**
   * @dev called by the owner to pause, triggers stopped state
   */
  function pause() onlyOwner whenNotPaused  public returns (bool) {
    paused = true;
    emit Pause();
    return true;
  }

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


/**
 * Pausable token
 *
 * Simple ERC20 Token example, with pausable token creation
 **/

contract PausableToken is StandardToken, Pausable {

  function transfer(address _to, uint _value) whenNotPaused  public {
    super.transfer(_to, _value);
  }
  function transferFrom(address _from, address _to, uint _value) whenNotPaused  public {
    super.transferFrom(_from, _to, _value);
  }
}



/**
 * @title TokenTimelock
 * @dev TokenTimelock is a token holder contract that will allow a
 * beneficiary to extract the tokens after a time has passed
 */
contract TokenTimelock {

  // ERC20 basic token contract being held
  ERC20Basic token;

  // beneficiary of tokens after they are released
  address public beneficiary;

  // timestamp where token release is enabled
  uint public releaseTime;

  constructor(ERC20Basic _token, address _beneficiary, uint _releaseTime)  public {
    require(_releaseTime > now);
    token = _token;
    beneficiary = _beneficiary;
    releaseTime = _releaseTime;
  }

  /**
   * @dev beneficiary claims tokens held by time lock
   */
  function claim()  public {
    require(msg.sender == beneficiary);
    require(now >= releaseTime);

    uint amount = token.balanceOf(this);
    require(amount > 0);

    token.transfer(beneficiary, amount);
  }
}
```

```
/**
 * @title FACTSToken
 * @dev Facts Token contract
 */
contract FactsToken is PausableToken, MintableToken {
  using SafeMath for uint256;

  string public name = "FACTS Token";
  string public symbol = "FACTS";
  uint public decimals = 18;
```

//SlowMist// 动态创建锁仓合约，并与挖矿(mint)融合在一起，值得推荐的模范

```
  /**
   * @dev mint timelocked tokens
   */
  function mintTimelocked(address _to, uint256 _amount, uint256 _releaseTime) public
    onlyOwner canMint returns (TokenTimelock) {

    TokenTimelock timelock = new TokenTimelock(this, _to, _releaseTime);
    mint(timelock, _amount);

    return timelock;
  }
```

//SlowMist// 预留映射所需的注册功能，思考很全面

```
  mapping (address => string) public  keys;
  event LogRegister (address user, string key);
  // Value should be a public key.  Read full key import policy.
  // Manually registering requires a base58
  // encoded using the STEEM, BTS, or EOS public key format.
  function register(string key) public {
      assert(bytes(key).length <= 64);
      keys[msg.sender] = key;
      emit LogRegister(msg.sender, key);
    }

  // If the user transfers ETH to contract, it will revert
  function () public payable{ revert(); }
```

```
}
```

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