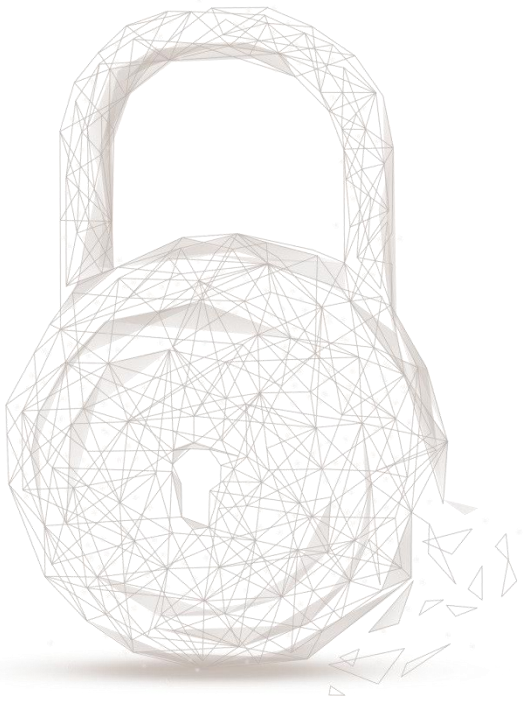
链安科技
Blockchain Security

Smart contract security audit report



Audit number : 201809030950

Smart Contract name :

Origo (OGO)

Smart Contract address :

0xFF0E5e014cf97e0615cb50F6f39Da6388E2FaE6E

Smart Contract Link address :

https://etherscan.io/address/0xff0e5e014cf97e0615cb50f6f39da6388e2fae6e#code

Start date of audit contract : 2018.08.09

Completion date of audit contract : 2018.09.03

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	Fake Deposit Audit	-	Pass
10	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 // It is recommended to use a fixed compiler version
// File: openzeppelin-solidity/contracts/ownership/Ownable.sol

/**
 * @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;


  event OwnershipRenounced(address indexed previousOwner);
  event OwnershipTransferred(
    address indexed previousOwner,
    address indexed newOwner
  );


  /**
   * @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 relinquish control of the contract.
  * @notice Renouncing to ownership will leave the contract without an owner.
  * It will not be possible to call the functions with the `onlyOwner`
  * modifier anymore.
  */
```

// Chengdu LianAn // Caution! Executing this function may cause owner losing the ownership of contract, and it will not be restorable.

```
  function renounceOwnership() public onlyOwner {
    emit OwnershipRenounced(owner);
    owner = address(0);
  }

  /**
  * @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) public onlyOwner {
    _transferOwnership(_newOwner);
  }

  /**
  * @dev Transfers control of the contract to a newOwner.
  * @param _newOwner The address to transfer ownership to.
  */
  function _transferOwnership(address _newOwner) internal {
    require(_newOwner != address(0));
    emit OwnershipTransferred(owner, _newOwner);
    owner = _newOwner;
  }
}

// File: openzeppelin-solidity/contracts/lifecycle/Pausable.sol

/**
 * @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 make a function callable only when the contract is not paused.
   */
  modifier whenNotPaused() {
    require(!paused);
    _;
  }

  /**
   * @dev Modifier to make a function callable only when the contract is paused.
   */
  modifier whenPaused() {
    require(paused);
    _;
  }

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

  /**
   * @dev called by the owner to unpause, returns to normal state
   */
  function unpause() onlyOwner whenPaused public {
    paused = false;
```

```
    emit Unpause();
  }
}

// File: openzeppelin-solidity/contracts/math/SafeMath.sol

/**
 * @title SafeMath
 * @dev Math operations with safety checks that throw on error
 */
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);
    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);
    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);
    return c;
  }
}

// File: openzeppelin-solidity/contracts/token/ERC20/ERC20Basic.sol

/**
 * @title ERC20Basic
 * @dev Simpler version of ERC20 interface
 * See https://github.com/ethereum/EIPs/issues/179
 */
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);
}

// File: openzeppelin-solidity/contracts/token/ERC20/BasicToken.sol

/**
 * @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));
    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;
  }

  /**
  * @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];
  }

}

// File: openzeppelin-solidity/contracts/token/ERC20/ERC20.sol

/**
 * @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
  );
}

// File: openzeppelin-solidity/contracts/token/ERC20/StandardToken.sol

/**
 * @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));
  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;
}

/**
  * @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.
  */
 // 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, function increaseApproval() and function decreaseApproval() are recommended to edit allowance.
 function approve(address _spender, uint256 _value) public returns (bool) {
  allowed[msg.sender][_spender] = _value;
  emit Approval(msg.sender, _spender, _value);
  return true;
 }

 /**
  * @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;
 }

}

// File: openzeppelin-solidity/contracts/token/ERC20/PausableToken.sol

/**
 * @title Pausable token
 * @dev StandardToken modified with pausable transfers.
 **/
// Chengdu LianAn // rewrite sensitive functions，add whenNotPaused modifier，so that owner can suspend all transactions when encountering big issues.
contract PausableToken is StandardToken, Pausable {

 function transfer(
  address _to,
  uint256 _value
 )
  public
  whenNotPaused
  returns (bool)
 {
  return super.transfer(_to, _value);
 }

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

```
  {
    return super.transferFrom(_from, _to, _value);
  }

  function approve(
    address _spender,
    uint256 _value
  )
    public
    whenNotPaused
    returns (bool)
  {
    return super.approve(_spender, _value);
  }

  function increaseApproval(
    address _spender,
    uint _addedValue
  )
    public
    whenNotPaused
    returns (bool success)
  {
    return super.increaseApproval(_spender, _addedValue);
  }

  function decreaseApproval(
    address _spender,
    uint _subtractedValue
  )
    public
    whenNotPaused
    returns (bool success)
  {
    return super.decreaseApproval(_spender, _subtractedValue);
  }
}
```

```
// File: contracts/OrigoToken.sol

contract OrigoToken is PausableToken {

    string public constant name = "Origo";
    string public constant symbol = "OGO";
    uint8 public constant decimals = 18;

    uint256 public constant INITIAL_SUPPLY = (10 ** 9) * (10 ** uint256(decimals));

    /**
    * @dev Constructor that gives msg.sender all of existing tokens.
    */
    constructor() public {
        totalSupply_ = INITIAL_SUPPLY;
        balances[msg.sender] = INITIAL_SUPPLY;
        emit Transfer(0x0, msg.sender, INITIAL_SUPPLY);
    }

}
```

链安科技

Blockchain Security

Official Website

https://lianantech.com

E-mail

vaas@lianantech.com

Twitter

https://twitter.com/LianAnTech