智能合约安全审计报告

慢雾安全团队于 2018-10-12 日， 收到 Hyperion Token 团队 Hyperion Token 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：

HYN

合约地址：

Hyperion.sol : 0xe99a894a69d7c2e3c92e61b64c505a6a57d2bc07

HyperionAirDrop.sol : 0xf8928417f49e520e201576cf34019059c427a75f

地址链接：

Hyperion.sol :

https://etherscan.io/address/0xe99a894a69d7c2e3c92e61b64c505a6a57d2bc07

HyperionAirDrop.sol:

https://etherscan.io/address/0xf8928417f49e520e201576cf34019059c427a75f

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	算数精度误差	-	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：0X001810140001

审计日期：2018 年 10 月 14 日

审计团队：慢雾安全团队

总结：此为代币(token)合约，不包含锁仓(tokenVault)部分。合约不存在溢出、条件竞争，综合评估（合约无风险）。

合约源代码如下：

```
pragma solidity ^0.4.24;

//SlowMist// 合约不存在溢出、条件竞争

//SlowMist// 使用了 OpenZeppelin 的 SafeMath 安全模块，值得称赞的做法


// 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/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/BasicToken.sol

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

  mapping(address => uint256) internal balances;

  uint256 internal 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(_value <= balances[msg.sender]);

    require(_to != address(0)); //SlowMist// 这类检查很好，避免用户失误导致 Token 转丢


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

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

  }

  /**
  * @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(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

    require(_to != address(0)); //SlowMist// 这类检查很好，避免用户失误导致 Token 转丢

    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// 返回值符合 EIP20 规范

  }
```

```
/**
  * @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// 返回值符合 EIP20 规范

}

/**
  * @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: contracts/Hyperion.sol

contract Hyperion is StandardToken {
    string  public constant name = "Hyperion Token";
    string  public constant symbol = "HYN";
    uint8   public constant decimals = 18;

    uint256 public constant INITIAL_SUPPLY = (10 * 1000 * 1000 * 1000) * (10 ** uint256(decimals));

    constructor() public {
        totalSupply_ = INITIAL_SUPPLY;
        balances[msg.sender] = INITIAL_SUPPLY;
        emit Transfer(address(0), msg.sender, INITIAL_SUPPLY);
    }
}
```

HyperionAirDrop.sol

```
pragma solidity ^0.4.24;
```

//SlowMist// 合约不存在溢出、条件竞争

//SlowMist// 使用了 OpenZeppelin 的 SafeMath 安全模块，值得称赞的做法

```
// 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/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.
 */
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)); //SlowMist// 这类检查很好，避免操作失误导致合约控制权丢失
    emit OwnershipTransferred(owner, _newOwner);
    owner = _newOwner;
  }
}

// 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) internal balances;

  uint256 internal 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(_value <= balances[msg.sender]);

    require(_to != address(0)); //SlowMist// 这类检查很好，避免用户失误导致 Token 转丢


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

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

  }

  /**
  * @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(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

    require(_to != address(0)); //SlowMist// 这类检查很好，避免用户失误导致 Token 转丢


    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// 返回值符合 EIP20 规范

}

/**
  * @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// 返回值符合 EIP20 规范

}

/**
  * @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: contracts/Hyperion.sol

contract Hyperion is StandardToken {
        string  public constant name = "Hyperion Token";
        string  public constant symbol = "HYN";
        uint8   public constant decimals = 18;

        uint256 public constant INITIAL_SUPPLY = (10 * 1000 * 1000 * 1000) * (10 ** uint256(decimals));

        constructor() public {
                totalSupply_ = INITIAL_SUPPLY;
                balances[msg.sender] = INITIAL_SUPPLY;
                emit Transfer(address(0), msg.sender, INITIAL_SUPPLY);
        }
}

// File: contracts/HyperionAirdrop.sol

contract HyperionAirdrop is Ownable {
        address public token;

        constructor(address _token) public {
                token = _token;
        }

        function remainingSupply() public view onlyOwner returns (uint256) {
                return Hyperion(token).balanceOf(owner);
        }

        function balanceOf(address _who) public view onlyOwner returns (uint256) {
```

```
        return Hyperion(token).balanceOf(_who);
    }

    function decimalFactor() internal view returns (uint256) {
        uint8 decimals = Hyperion(token).decimals();
        return (10 ** uint256(decimals));
    }

    function transfer(address _to, uint256 _value) public onlyOwner returns (bool) {

        uint256 _number = SafeMath.mul(_value, decimalFactor());
        return Hyperion(token).transferFrom(owner, _to, _number);
    }

    function batchTransfer(address[] _recipients, uint256[] _values) public onlyOwner returns (bool) {
        require(_recipients.length == _values.length);

        for(uint256 i = 0; i< _recipients.length; i++)
        {
            address _to = _recipients[i];
            uint256 _value = _values[i];

            require(transfer(_to, _value));
        }


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

    }

    function destroy() public onlyOwner {
        selfdestruct(owner);
    }
}
```

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