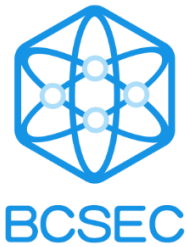
DVP TOKEN

智能合约安全审计报告



目录

声明

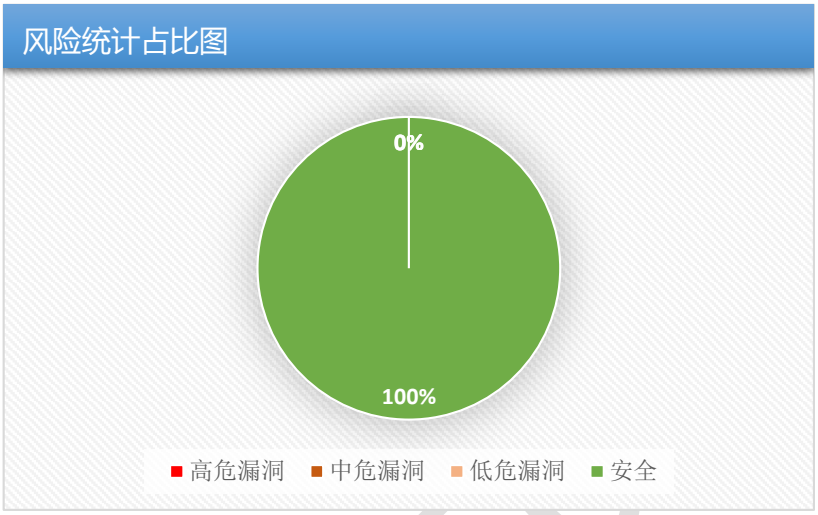
本审计报告采用形式化验证的方式来尽可能多地发现该项目中存在的问题，但并不能代表在以后不会出现新型问题。BCSEC 在任何情况下都推荐采用多方、多次独立审计的方式来从不同维度来保障区块链项目的安全。

本审计报告仅对该项目的安全性进行客观分析，不构成任何投资建议。

审计日期：2018 年 11 月 2 日

结果概览

该版本的 DVP Token 智能合约无严重漏洞，整体安全等级相对较高。



表格 1风险统计占比图

风险	危害性	结果
可欺诈隐患	低危	安全
数值溢出隐患	严重	安全
重入漏洞隐患	高危	安全
访问控制缺陷隐患	高危	安全
跨合约调用漏洞	高危	安全
拒绝服务漏洞漏洞	高危	安全
特权账户隐患	低危	安全
矿工特权隐患	低危	安全
随机值可预测隐患	低危	安全
设计不当隐患	高危	安全
逻辑漏洞隐患	高危	安全

表格 2 检测项表

检测项

可欺诈隐患—安全

可欺诈隐患的检测条目主要包括：特权函数可滥用、tx.Origin 钓鱼攻击、短地址攻击、合约后门、白皮书与代币实际发行总量不一致等有可能对投资者带来资金风险的安全隐患。

经 BCSEC 检测，检测结果为"**通过**"，原因如下：

- 特权函数滥用检测：经检测，合约中的特权函数仅有 3 个，分别是 transferOwnership 函数、pause 函数、unpause 函数，除此之外并无其他特权函数。

```
45  function transferOwnership(address _newOwner) public onlyOwner {
46    require(_newOwner != address(0));
47    emit OwnershipTransferred(owner, _newOwner);
48    owner = _newOwner;
49  }
```

transferOwnership 函数解析：该函数的作用主要是用于转让 Onwer 权限，并无其他风险。

```
57  modifier whenNotPaused() {
58    require(!paused);
59    _;
60  }
61
62  modifier whenPaused() {
63    require(paused);
64    _;
65  }
66
67  function pause() onlyOwner whenNotPaused public {
68    paused = true;
69    emit Pause();
70  }
71
72  function unpause() onlyOwner whenPaused public {
73    paused = false;
74    emit Unpause();
75  }
```

pause 函数、unpause 函数解析：这两个函数的作用主要用于暂停 Token 交易，其功能是为了在主网上线时暂停 Token 的使用，避免主网上线后一些用户还在使用 Token 进行交易而遭受损失，无其他安全风险。

- tx.Origin 钓鱼攻击检测：经检测，合约中的没有使用 tx.Origin 变量，不存在此类风险。

- 短地址攻击检测：经检测，合约在进行转账、资金授权的时候都有检测 msg.data.length >= (2 * 32) + 4，可有效避免短地址攻击。

```
123   function transfer(address _to, uint256 _value) public returns (bool success) {
124       require(msg.data.length >= (2 * 32) + 4);
125       require(_to != address(0));
126       require(_value <= balances[msg.sender]);
127
128       balances[msg.sender] = balances[msg.sender].sub(_value);
129       balances[_to] = balances[_to].add(_value);
130       emit Transfer(msg.sender, _to, _value);
131       return true;
132
133   }
```

```
151   function approve(address _spender, uint256 _value) public returns (bool success) {
152       require(msg.data.length >= (2 * 32) + 4);
153       require(_value == 0 || allowed[msg.sender][_spender] == 0);
154       allowed[msg.sender][_spender] = _value;
155       emit Approval(msg.sender, _spender, _value);
156       return true;
157   }
```

- 白皮书与代币实际发行总量不一致检测：经检测，代币合约中的 totalSupply 与白皮书声明的发行总量一致，同为 50 亿，不存在此类风险。

数值溢出隐患—安全

数值溢出隐患的检测条目主要包括：是否对加减乘除等各类运算进行溢出检测，避免超发代币等风险的安全隐患。

经 BCSEC 检测，检测结果为"**通过**"，原因如下：

- 算术溢出检测：经检测，合约中对所有的数值运算均采用了 SafeMath 库，没有任何一处使用了原生的加减乘除，不存在算术溢出的风险。

```
library SafeMath {
 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
   if (a == 0) {
     return 0;
   }
   uint256 c = a * b;
   assert(c / a == b);
   return c;
   }

  function div(uint256 a, uint256 b) internal pure returns (uint256) {
   uint256 c = a / b;
   return c;
  }

 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
   assert(b <= a);
   return a - b;
  }

  function add(uint256 a, uint256 b) internal pure returns (uint256) {
   uint256 c = a + b;
   assert(c >= a);
   return c;
  }
```

```
contract BasicToken is ERC20Basic {
  using SafeMath for uint256;
  mapping(address => uint256) balances;
  uint256 totalSupply_;
```

重入漏洞隐患—安全

重入漏洞隐患的检测条目主要包括：是否使用 transfer 函数转移 Ether、是否在转账等敏感操作之前完成状态更新。

经 BCSEC 检测，检测结果为"**通过**"，原因如下：

- 使用 transfer 函数转移 Ether 检测：经检测，合约中并没有涉及到 Ether 转移的操作，不存在该类的风险。

- 在转账等敏感操作之前完成状态更新检测：经检测，合约中并没有涉及到 Ether 转移的操作，不过存在 Token 转账操作，但 Token 转账操作无法触发调用外部合约，所以不存在该类的风险。

访问控制缺陷隐患—安全

访问控制缺陷隐患的检测条目主要包括：特权函数访问绕过、是否存在"伪构造函数"。
经 BCSEC 检测，检测结果为"**通过**"，原因如下：

- 特权函数访问绕过检测：经检测，合约中对仅有的 3 个特权函数都使用了 onlyOwner 修饰符进行鉴权，且 onlyOwner 修饰符无绕过问题，不存在该类风险。

- "伪构造函数"函数检测：经检测，合约采用 solidity 0.4.24 进行编译，使用 constructor 关键字声明构造函数，该版本的编译器支持 constructor 关键字来进行构造函数声明，有效地避免了构造函数笔误造成的访问控制缺陷等问题，不存在该类风险。

跨合约调用隐患—安全

跨合约调用隐患的检测条目主要包括：Call 函数注入。
经 BCSEC 检测，检测结果为"**通过**"，原因如下：

- Call 函数注入检测：经检测，合约中在 approveAndCall 函数中使用了 Call 函数，但是 Call 函数调用的外部合约函数是指定的，无法造成注入，所以不存在该类风险。

```
221     function approveAndCall(address _spender, uint256 _value, bytes _extraData) public returns (bool success) {
222         allowed[msg.sender][_spender] = _value;
223         emit Approval(msg.sender, _spender, _value);
224         if(!_spender.call(bytes4(bytes32(keccak256("receiveApproval(address,uint256,address,bytes)"))), msg.sender, _value, this, _extraData)) { revert(); }
225         return true;
226     }
227 }
```

拒绝服务漏洞隐患—安全

拒绝服务漏洞隐患的检测条目主要包括：Out of gas 风险、依赖库可销毁风险、"暂停"函数滥用风险。
经 BCSEC 检测，检测结果为"**通过**"，原因如下：

- Out of gas 风险检测：经检测，合约中没有使用 for、while 等消耗大量 Gas 的关键字，在现有的函数中，每个函数所能消耗的 Gas 都是在可控范围之内的，所以不存在该类风险。

- 依赖库可销毁风险检测：经检测，合约中使用了 SafeMath 这个依赖库，但该依赖库不存在可导致销毁的函数，所以不存在该类风险。

- "暂停"函数滥用风险检测：经检测，合约中使用了 pause 和 unpause 函数来控制合约代币是否可交易，但这两个函数都经过了严格的鉴权，只能由 owner 调用，所以不存在该类风险。

特权账户隐患—安全

特权账户隐患的检测条目主要包括：特权账户的权限是否在允许范围之内。
经 BCSEC 检测，检测结果为"**通过**"，原因如下：

- 特权账户权限检测：经检测，合约中特权账户所的特权有暂停交易、开放交易，除此之外没有其他可以影响用户资金、任意增发等特权，所以不存在该类风险。

矿工特权隐患—安全

矿工特权隐患的检测条目主要包括：区块数据依赖、事务顺序依赖。
经 BCSEC 检测，检测结果为"**通过**"，原因如下：

- 区块数据依赖检测：经检测，合约中没有依赖区块中的任何数据，所以不存在该类风险。

- 区块数据依赖检测：经检测，合约中的 approve 函数存在事务顺序依赖，但该函数中限定了授权额度被耗尽或者将授权额度归零时才能调用该函数，一定程度上避免了事务顺序依赖带来的问题，而且合约中还使用了 increaseApproval 函数和 decreaseApproval 函数来作为补充方案，所以不存在该类风险。

随机值可预测隐患—安全

随机值隐患的检测条目主要包括：随机值生成算法。
经 BCSEC 检测，检测结果为"**通过**"，原因如下：

- 随机值生成算法检测：经检测，合约中没有涉及到使用随机值的相关函数，所以不存在该类风险。

设计不当隐患—安全

设计不当隐患的检测条目主要包括：合约设计是否遵循相关标准、合约设计是否存在"薅羊毛"等问题。

经 BCSEC 检测，检测结果为"**通过**"，原因如下：

- 合约设计是否遵循相关标准检测：经检测，该合约严格遵循 ERC20 标准来进行设计以及实现，所以不存在该类风险。
- 合约设计是否存在"薅羊毛"等问题检测：经检测，该合约除了 ERC20 实现以及其他常规函数，没有设计其他例如"airdrop"等额外函数，所以不存在该类风险。

逻辑漏洞隐患—安全

逻辑漏洞隐患的检测条目主要包括：各函数是否存在逻辑缺陷、各修饰符是否存在逻辑缺陷。

经 BCSEC 检测，检测结果为"**通过**"，原因如下：

- 各函数是否存在逻辑缺陷检测：经检测，该合约的各函数逻辑都相对简单，且都进行了一致性校验、身份校验等，在业务逻辑上没有明显缺陷，不存在该类风险。

- 各修饰符是否存在逻辑缺陷检测：经检测，该合约的各修饰符都符合正常业务逻辑，且不存在例如把"=="写成"！ ="等笔误造成的逻辑缺陷，不存在该类风险。

漏洞详情

无明显漏洞

描述：

该合约在整体上安全性较高，无明显安全问题。

缺陷代码片段：

无

危害说明：

无

防护：

无

建议

该合约在各方面皆符合要求，且没有明显缺陷，无建议。

总结

在审计该合约时，BCSEC 团队注意到该合约使用的是 ConsenSys 所提供的标准 ERC20 模板，此模板安全强度非常之高，两年之内没有修改过模板代码，也从未出现过问题。

同时，该合约还使用了 SafeMath 库来杜绝整型溢出的问题。

在特权隐患方面，该合约直接了设立 owner 角色，但 owner 角色的特权并不大,大大减小了攻击面以及杜绝了特权账户作恶的隐患。

越是简单便越是安全，该合约中用户可写数据能力的函数仅有 9 个，攻击面非常之小，安全性很高。

附录 A:关于 DVPToken

DVP Token 定义

DVP Token 是 DVP 生态系统网络内唯一被接受的货币。永不增发，目前任何 DVP 网络的支付，悬赏，押金质押，必须用 DVP 币结算。DVP 币支付用于奖励社区成员，为他们提供资金流动。

总量：5,000,000,000

符号：DVP

DVP Token 用途

用于厂商发布悬赏的押金；

用于支付漏洞奖励；

用于维持生态运营，如开发，宣传推广等奖励；

ERC20 设计说明

支持锁仓：按照白皮书中设计的代币分配将一定量的代币进行锁仓，按季度等额释放。

权限弱化：代币中设有 owner 账户，但是 owner 账户只有暂停交易的特权（主网上线后使用），无法对用户资产造成任何影响，杜绝单点威胁。

高度安全：代币合约已经经过了多家专业区块链安全公司审计并加固，具有极高的安全强度。

总量恒定：DVP Token 没有设计代币销毁/增发功能，所以代币的总量是固定的。

附录 B:关于 BCSEC

BCSEC(https://bcsec.org/)专注聚焦区块链安全生态，提供区块链行业领先的安全解决方案。我们关注和研究最前沿的漏洞以及相关安全情报资讯，目前已对数字钱包、交易所、矿池、智能合约等多个应用场景有深厚研究积累，可为区块链社区安全运营提供预警，持续为区块链安全生态提供技术支撑。