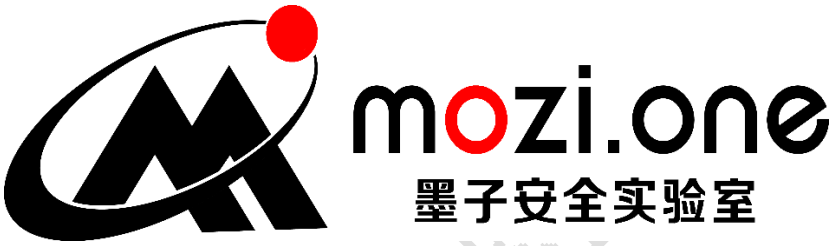


FST 智能合约安全审计报告

（敏感信息，注意保密）

墨子安全实验室

智能合约审计组

2019 年 5 月 6 日

■ 法律声明

1. 本文档为智能合约项目的安全审计报告，供项目研发部门参考使用，本文档包含双方商业和技术秘密，严禁外泄。任何个人、机构未经墨子安全实验室的书面授权许可，不得以任何方式复制或引用本文的任何片断。

2. 该项目的安全审查工作需要得到项目方的密切配合，以便墨子安全实验室对项目的背景、架构和流程有充分的理解，由于项目方不配合、误导或者信息错误引起的一切后果，墨子安全实验室不承担责任。

3. 安全审计工作只能报告出有限的缺陷和漏洞，具有不可消除的误报率和漏报率。同时，缺陷和漏洞不等同于漏洞利用，从缺陷和漏洞到被利用产生后果还有一段不可预测的距离。

4. 墨子安全实验室不对该项目的运营安全问题负责，对于因为运营安全问题引发的损失（包括但不限于黑客攻击等）不承担任何责任。

版本修订：

编号	时间	修订模式	修订者	修订内容
1	5/1/2019	新建	墨子安全实验室 智能合约审计组	新建
2	5/6/2019	修改	苗知秋	审定
3				
4				
5				
6				
7				

备注：修订模式包括新建、增加、删除、修改等

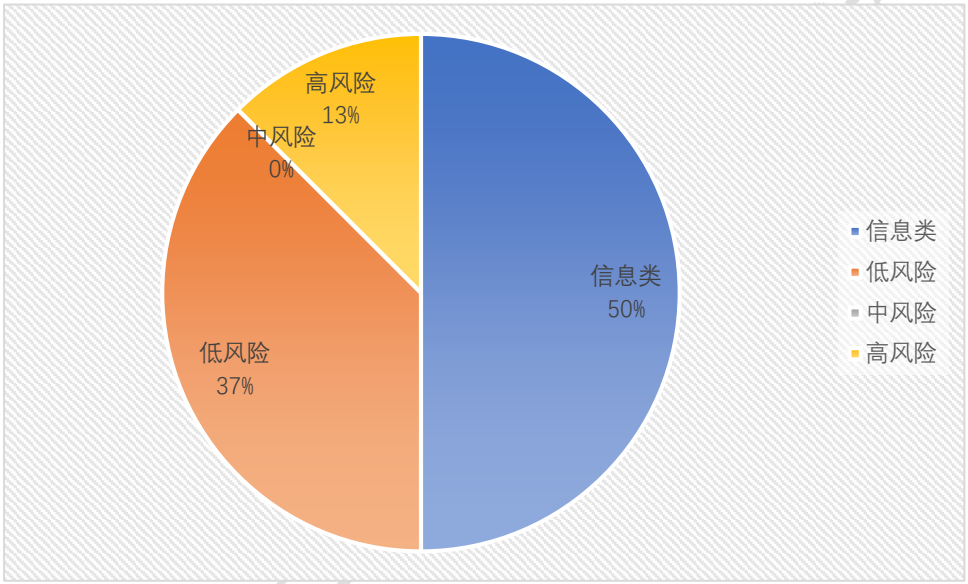
目录

1 报告概要

此次审计中，我们对审计对象的通用安全风险、子链安全风险、一致性进行了检测分析。

➢ 通用安全风险

发现 3 个低风险问题，1 个高风险问题，4 个信息类问题，如下图所示：



根据附录 B 的智能合约安全风险状况等级说明以及威链交易所信息安全管控规则，本审计对象处于：

预警状态

详情见本报告“审计结果详情”。

2 审计时间

本次审计时间（包括审计测试和报告撰写）从 2019 年 5 月 5 日开始到 5 月 6 日结束，在此期间针对如下审计对象进行测试并以此作为本报告依据。

3 审计对象

3.1 项目源代码

包括：

	文件名/地址	SHA256值
1	FST_Token_v0.1.sol	05ac6c6fc8e6d69ec22bcb75f94019ad81f65990cc628cfc4e411db76cfbb30d

3.2 项目功能描述文档

无

3.3 项目需求描述文档

无

3.4 项目测试/部署文档

无

4 审计内容

本次审计内容为检测审计对象是否具有通用安全风险。

5 审计方法

对本项目的测评采用自动化工具和人工分析相结合的方式。自动化工具包括采用静态分析工具和动态分析工具，对输出结果经过人工审核分析确认。人工分析设计文档，对设计逻辑进行风险评估。对发现的问题，在可能的情况下模拟攻击进行渗透测试，以证实漏洞的真实性和风险等级。

6 审计对象描述

FST_Token_v0.1.sol 实现的代币合约 FSTToken 基本的 ERC20 合约（https://eips.ethereum.org/EIPS/eip-20） 功能增加了冻结功能。

```
1   /**
2    * Overflow aware uint math functions.
3    *
4    * Inspired by https://github.com/MakerDAO/maker-otc/blob/master/contracts/simple_market.sol
5    */
6   pragma solidity ^0.4.11;
7
8   /**
9    * ERC 20 token
10   *
11   * https://github.com/ethereum/EIPs/issues/20
12   */
13  contract FSTToken {
14      string public constant name = "FST Token";
15      string public constant symbol = "FST";
16      uint public constant decimals = 18;
17      uint256 _totalSupply = 2000000000 * 10**decimals;
18      uint256 miningCap = 1200000000 * 10**decimals;
19      uint256 minedSoFar = 0;
20      uint256 teamCap = 300000000 * 10**decimals;
21      uint256 teamSoFar = 0;
22      uint256 stakeCap = 200000000 * 10**decimals;
23      uint256 stakeSoFar = 0;
24      uint256 foundationCap = 200000000 * 10**decimals;
25      uint256 foundationSoFar = 0;
26      uint256 dappCap = 100000000 * 10**decimals;
27      uint256 dappSoFar = 0;
28
29
30      function totalSupply() public view returns (uint256 supply) {
31          return _totalSupply;
32      }
33
34      function balanceOf(address _owner) public view returns (uint256 balance) {
35          return balances[_owner];
36      }
37
38      function approve(address _spender, uint256 _value) public returns (bool success) {
```

```
39                  allowed[msg.sender][_spender] = _value;
40                  Approval(msg.sender, _spender, _value);
41                  return true;
42              }
43
44              function allowance(address _owner, address _spender) public returns (uint256 remaining) {
45                  return allowed[_owner][_spender];
46              }
47
48              mapping(address => uint256) balances; //list of balance of each address
49              mapping(address => uint256) distBalances; //list of distributed balance of each address to
calculate restricted amount
50              mapping(address => uint256) teamBalances;
51              mapping(address => uint256) dappBalances;
52              mapping(address => mapping (address => uint256)) allowed;
53
54              mapping(address => bool) dAppAccounts;
55              mapping(address => bool) teamAccounts;
56
57    //All other time spots are calculated based on this time spot.
58                uint public baseStartTime = now;
59
60              // Initial founder address (set in constructor)
61              // All deposited MOAC will be instantly forwarded to this address.
62              address public founder = 0xd3183c78c35a884bd544165e288b85e732227ca0;
63              address public admin = 0x22c139608d92ec8017aab057b06d0abafe53da98;
64    address public lockAddress = 0x0;
65    uint lockKillTime = 0;
66    bool lockFlag = true;
67    bool timeLock = true;
68
69              uint256 public distributed = 0;
70
71              event AllocateFounderTokens(address indexed sender);
72              event Transfer(address indexed _from, address indexed _to, uint256 _value);
73              event Approval(address indexed _owner, address indexed _spender, uint256 _value);
74
75              //constructor
76              function FSTToken() public {
77              }
78
79              function setStartTime(uint _startTime) public {
80                  if (msg.sender!=founder && msg.sender!=admin && timeLock == true) revert();
81      if (now > _startTime) revert();
```

```
82                  baseStartTime = _startTime;
83      timeLock = false;
84              }
85
86      function setLockAddress(address lock) public {
87                  if (msg.sender!=founder && msg.sender!=admin) revert();
88      if (lockFlag == false) revert();
89                  lockAddress = lock;
90      lockFlag = false;
91      lockKillTime = now + 86400;
92              }
93
94              function distributeMined(uint256 _amount, address _to) public {
95                  if (msg.sender!=founder && msg.sender!=admin) revert();
96                  if (distributed + _amount > _totalSupply) revert();
97                  if (minedSoFar + _amount > miningCap) revert();
98
99                  distributed += _amount;
100                   minedSoFar += _amount;
101
102                   balances[_to] += _amount;
103                   distBalances[_to] += _amount;
104             }
105
106             function distributeFoundation(uint256 _amount, address _to) public {
107                 if (msg.sender!=founder && msg.sender!=admin) revert();
108                 if (distributed + _amount > _totalSupply) revert();
109                 if (foundationSoFar + _amount > foundationCap) revert();
110
111                 distributed += _amount;
112                 foundationSoFar += _amount;
113
114                 balances[_to] += _amount;
115                 distBalances[_to] += _amount;
116             }
117
118             function distributeStake(uint256 _amount, address _to) public {
119                 if (msg.sender!=founder && msg.sender!=admin) revert();
120                 if (distributed + _amount > _totalSupply) revert();
121                 if (stakeSoFar + _amount > stakeCap) revert();
122
123                 distributed += _amount;
124                 stakeSoFar += _amount;
125
```

```
126                 balances[_to] += _amount;
127                 distBalances[_to] += _amount;
128             }
129
130             function distributeTeam(uint256 _amount, address _to) public {
131                 if (msg.sender!=founder && msg.sender!=admin) revert();
132                 if (distributed + _amount > _totalSupply) revert();
133                 if (teamSoFar + _amount > teamCap) revert();
134
135                 distributed += _amount;
136                 teamSoFar += _amount;
137
138                 balances[_to] += _amount;
139                 distBalances[_to] += _amount;
140                 teamBalances[_to] += _amount;
141             }
142
143             function distributeDapp(uint256 _amount, address _to) public {
144                 if (msg.sender!=founder && msg.sender!=admin) revert();
145                 if (distributed + _amount > _totalSupply) revert();
146                 if (dappSoFar + _amount > dappCap) revert();
147
148                 distributed += _amount;
149                 dappSoFar += _amount;
150
151                 balances[_to] += _amount;
152                 distBalances[_to] += _amount;
153                 dappBalances[_to] += _amount;
154             }
155
156             function transfer(address _to, uint256 _value) public returns (bool success) {
157
158                 //Default assumes totalSupply can't be over max (2^256 - 1).
159                 //If your token leaves out totalSupply and can issue more tokens as time goes on, you need
to check if it doesn't wrap.
160                 //Replace the if with this one instead.
161                 if (balances[msg.sender] >= _value && balances[_to] + _value > balances[_to]) {
162                     uint _freeAmount = freeAmount(msg.sender);
163                     if (_freeAmount < _value) {
164                         return false;
165                      }
166
167                     balances[msg.sender] -= _value;
168                     balances[_to] += _value;
```

```
169                     Transfer(msg.sender, _to, _value);
170                     return true;
171                 } else {
172                     return false;
173                 }
174             }
175
176             function freeAmount(address user) public returns (uint256 amount) {
177                 //1) no free amount before base start time;
178                 if (now < baseStartTime) {
179                     return 0;
180                 }
181
182                 //2) calculate number of months passed since base start time;
183                 uint monthDiff = (now - baseStartTime) / (30 days);
184
185                 //3) if it is over 18 months, free up everything.
186                 if (monthDiff > 18) {
187                     return balances[user];
188                 }
189
190                 //4) calculate amount of unrestricted within distributed amount.
191                 uint unfrozen = 0;
192                 if (monthDiff > 6) {
193                     if (balances[user] >= teamBalances[user]) {
194                         unfrozen = balances[user] - teamBalances[user];
195                     }
196                     return unfrozen;
197                 } else {
198                     if (balances[user] >= teamBalances[user]) {
199                         unfrozen = balances[user] - teamBalances[user];
200                     }
201                     if (unfrozen >= dappBalances[user]) {
202                         unfrozen = unfrozen - dappBalances[user];
203                     }
204                     return unfrozen;
205                 }
206             }
207
208             /**
209              * Change founder address (where FST is being forwarded).
210              *
211              * Applicable tests:
212              *
```

```
213             * - Test founder change by hacker
214             * - Test founder change
215             * - Test founder token allocation twice
216             */
217             function changeFounder(address newFounder) public {
218                 if (msg.sender!=founder) revert();
219                 founder = newFounder;
220             }
221
222             function changeAdmin(address newAdmin) public {
223                 if (msg.sender!=founder && msg.sender!=admin) revert();
224                 admin = newAdmin;
225             }
226
227             /**
228              * ERC 20 Standard Token interface transfer function
229              *
230              * Prevent transfers until freeze period is over.
231              */
232             function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
{
233                 if (msg.sender != founder && msg.sender!=admin) revert();
234
235                 //same as above. Replace this line with the following if you want to protect against
wrapping uints.
236                 if (balances[_from] >= _value && allowed[_from][msg.sender] >= _value &&
balances[_to] + _value > balances[_to]) {
237                 //if (balances[_from] >= _value && allowed[_from][msg.sender] >= _value && _value >
0){
238                     uint _freeAmount = freeAmount(_from);
239                     if (_freeAmount < _value) {
240                         return false;
241                      }
242
243                     balances[_to] += _value;
244                     balances[_from] -= _value;
245                     allowed[_from][msg.sender] -= _value;
246                     Transfer(_from, _to, _value);
247                     return true;
248                 } else { return false; }
249             }
250
251             function() public payable {
252                  if (!founder.call.value(msg.value)()) revert();
```

```
253             }
254
255             // only lockAddress can kill within one day after lockAddress is set
256             function kill() public {
257                 if (msg.sender == lockAddress && now < lockKillTime) {
258                     suicide(founder);
259                 }
260             }
261
262         }
```

7 审计结果详情

7.1 代码安全

7.1.1 版本配置

1. Use Latest Version

【说明】使用最新的 Release 编译器版本,而非过时的编译器版本。新的编译器版本在编译生成合约字节码时会引入比较少的漏洞。尤其 0.5.0 以后的版本，在安全性方面有了很大提升。

【风险级别】低

【审计结果】本审计对象使用了过时的编译器版本。

```
6       pragma solidity ^0.4.11;
```

【修改建议】使用最新的编译器版本编译本合约。

如部署环境不支持较新的编译器版本，可使用支持较新编译器版本的开发工具，如 Remix，编译生成字节码后，将字节码部署到区块链上。

7.1.2 整型计算

2. Check Overflow Underflow

【说明】整型变量如果超过了最大值，如 uint 超过 2^256-1，会上溢出为 0，如果小于 0，会下溢出为最大值（2^256-1）。如果发生溢出的变量和合约的金额相关，会引起严重后果。

【风险级别】低

【审计结果】本审计对象存在溢出风险。但因为可能出现溢出的函数限制了调用者为合约的创

建者或 admin，风险级别较低。

```
94          function distributeMined(uint256 _amount, address _to) public {
95              if (msg.sender!=founder && msg.sender!=admin) revert();
96              if (distributed + _amount > _totalSupply) revert();
97              if (minedSoFar + _amount > miningCap) revert();
98
99              distributed += _amount;
100             minedSoFar += _amount;
101
102             balances[_to] += _amount;
103             distBalances[_to] += _amount;
104         }
105
106         function distributeFoundation(uint256 _amount, address _to) public {
107             if (msg.sender!=founder && msg.sender!=admin) revert();
108             if (distributed + _amount > _totalSupply) revert();
109             if (foundationSoFar + _amount > foundationCap) revert();
110
111             distributed += _amount;
112             foundationSoFar += _amount;
113
114             balances[_to] += _amount;
115             distBalances[_to] += _amount;
116         }
117
118         function distributeStake(uint256 _amount, address _to) public {
119             if (msg.sender!=founder && msg.sender!=admin) revert();
120             if (distributed + _amount > _totalSupply) revert();
121             if (stakeSoFar + _amount > stakeCap) revert();
122
123             distributed += _amount;
124             stakeSoFar += _amount;
125
126             balances[_to] += _amount;
127             distBalances[_to] += _amount;
128         }
129
130         function distributeTeam(uint256 _amount, address _to) public {
131             if (msg.sender!=founder && msg.sender!=admin) revert();
132             if (distributed + _amount > _totalSupply) revert();
133             if (teamSoFar + _amount > teamCap) revert();
134
135             distributed += _amount;
136             teamSoFar += _amount;
```

```
137
138                 balances[_to] += _amount;
139                 distBalances[_to] += _amount;
140                 teamBalances[_to] += _amount;
141             }
142
143             function distributeDapp(uint256 _amount, address _to) public {
144                 if (msg.sender!=founder && msg.sender!=admin) revert();
145                 if (distributed + _amount > _totalSupply) revert();
146                 if (dappSoFar + _amount > dappCap) revert();
147
148                 distributed += _amount;
149                 dappSoFar += _amount;
150
151                 balances[_to] += _amount;
152                 distBalances[_to] += _amount;
153                 dappBalances[_to] += _amount;
154             }
```

【修改建议】对加法、乘法做溢出判断，或通过其他途径确保不出现溢出。

7.1.3 权限控制

3. Invalid Access Control

【说明】函数的权限控制失效或不符合常理。

【风险级别】高

【审计结果】本审计对象 L80 存在可能失效的权限控制。

```
79          function setStartTime(uint _startTime) public {
80              if (msg.sender!=founder && msg.sender!=admin && timeLock == true) revert();
81  if (now > _startTime) revert();
82              baseStartTime = _startTime;
83  timeLock = false;
84          }
```

【修改建议】权限控制改成 require(msg.sender==founder || msg.sender==admin)；

Require(timeLock == true);

4. Uint/int 变量类型

【说明】uint（int）是 uint256（int256）的别名，并不是 8 比特的 uint8（int8），为避免混乱，建议不使用别名。

【风险级别】信息

【审计结果】合约中同时存在声明为 uint 和 uint256 的变量，说明编码者认为这两者是有区别的，事实上 uint 是 uint256 的别名。

```
16          uint public constant decimals = 18;
17          uint256 _totalSupply      = 2000000000 * 10**decimals;
```

```
18      uint256 miningCap          = 1200000000 * 10**decimals;
19      uint256 minedSoFar         = 0;
20      uint256 teamCap            =  300000000 * 10**decimals;
21      uint256 teamSoFar          = 0;
22      uint256 stakeCap       =  200000000 * 10**decimals;
23      uint256 stakeSoFar     = 0;
24      uint256 foundationCap      =  200000000 * 10**decimals;
25      uint256 foundationSoFar = 0;
```

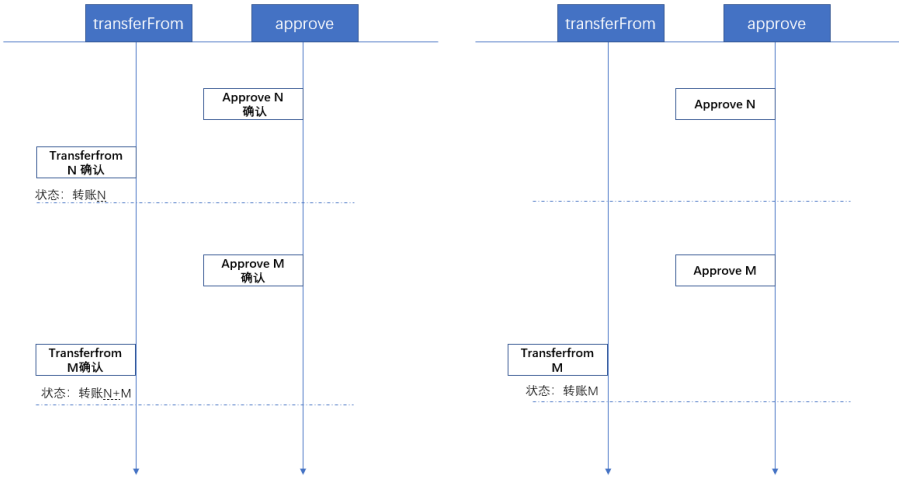
【修改建议】根据变量的长度需求明确要声明的变量类型：uint8,……,uint256。

7.2 设计逻辑

7.2.1 交易顺序

5. Avoid Approve Race Condition

【说明】ERC 20 中的 approve 函数如使用不谨慎，可能导致业务逻辑漏洞，引起 front running attack，如授权修改后，新旧授权均被执行。授权转账者执行调用 transferFrom 时需要授权者授权一定配额。当出现授权者通过 Approve 两次授权时，比如第一次授权 M，后修改至 N，而其本意是只授权 N，授权转账者监控到授权配额改变时，可抢先在第二次授权被确认前，先执行 M 的授权转账，等到第二次授权确认后，还可再次执行 N 的授权转账，造成超额授权转账。如下图所示：



【风险级别】低

【审计结果】本审计对象合约使用了函数 approve，没做安全加强，有竞争条件的风险。

```
38      function approve(address _spender, uint256 _value) public returns (bool success) {
39          allowed[msg.sender][_spender] = _value;
40          Approval(msg.sender, _spender, _value);
41          return true;
42      }
```

【修改建议】 approve 中增加先判断当前的授权额度是否为 0 或新授权额度为 0 的语句，如：require((_value == 0) || (allowed[msg.sender][_spender] == 0));

7.3 应用规范

7.3.1 参数检查

6. Transfer Zero

【说明】transfer，transferFrom 函数将 0 的交易看作正常交易，且发送事件。

【风险级别】信息

【审计结果】本审计对象的合约将 0 排除在外。

```
161                 if (balances[msg.sender] >= _value && balances[_to] + _value > balances[_to]) {
236                 if (balances[_from] >= _value && allowed[_from][msg.sender] >= _value &&
balances[_to] + _value > balances[_to]) {
```

【修改建议】不排除 0 的交易，修改建议成 require(balances[_to] + _value >= balances[_to]);。

7.3.2 事件处理

7. Transfer Event

【说明】ERC20（https://eips.ethereum.org/EIPS/eip-20）标准规定造币发生时需要触发发送者为 0 的 Transfer 事件，虽然没有明确规定 ERC20 合约初始化时应否也触发这种事件，但是凡和代币增加或减少相关的交易最好均触发 Transfer 事件，以便监控异常。

【风险级别】信息

【审计结果】本审计对象的合约在初始化时未触发 Transfer 事件。

【修改建议】在初始化函数中触发 Transfer（0，msg.sender, totalSupply）事件。

7.3.3 其他

8. Decimals be uint8

【说明】ERC20（https://eips.ethereum.org/EIPS/eip-20）标准规定 decimals 应为 uint8 类型。

```
function decimals() public view returns (uint8)
```

【风险级别】信息

【审计结果】本合约的 decimals 为 uint256。

```
16          uint public constant decimals = 18;
```

【修改建议】给 Transfer 函数增加返回值。

附录 A. 审计条目风险级别说明

由于智能合约的漏洞模型还没有共识，在此根据 CNVD 的漏洞分级规范，对合约的漏洞分级如下：

级别	风险描述	安全建议
极高	该漏洞可利用性高和后果影响度综合评定高。对攻击者要求不高，有公开可利用的工具，有的甚至发生过攻击事件先例。	立即修复
高	该漏洞可利用性高和后果影响度综合评定较高。可能导致系统权限被攫取或危害数据安全。	强烈建议修复
中	该漏洞可利用性和后果影响度综合评定中等。当和其他漏洞或攻击方式结合可能危害系统权限或数据。	建议修复
低	该漏洞可利用性和后果影响度综合评定较低。即使和其他漏洞或攻击方式结合也不会立即危害系统的权限和数据。	无
信息	非漏洞信息，包括编程规范、应用规范、代码优化等方面的改善建议信息。	无

注意： 由于智能合约的特殊性，如代码公开、基本不可修补，而且往往涉及有价代币，强烈建议只要是发现的问题，均在部署前进行修复。该表和条目中的分级仅供在修复时间计划上给予参考。

附录 B. 智能合约安全风险状况等级说明

安全风险状况说明	
良好状态	信息系统处于良好运行状态，没有发现或只存在零星的低风险安全问题，此时只要保持现有安全策略就满足了本系统的安全等级要求。
预警状态	信息系统中存在一些漏洞或安全隐患，此时需根据评估中发现的网络、主机、应用和管理等方面的问题对进行有针对性的加固或改进。
严重状态	信息系统中发现存在严重漏洞或可能严重威胁到系统正常运行的安全问题，此时需要立刻采取措施，例如安装补丁或重新部署安全系统进行防护等等。
紧急状态	信息系统面临严峻的网络安全态势，对组织的重大经济利益或政治利益可能造成严重损害。此时需要与其他安全部门通力协作采取紧急防御措施。

附录 C. 墨子安全实验室简介及安全服务体系

墨子安全实验室（曾用名：白墨子安全实验室）是一家专业的区块链安全实验室，2018 年 5 月成立于北京市。

针对区块链产业日益迫切的安全需求，墨子安全实验室基于“能攻善守，攻防一体”的技术定位，坚持“面向产业前沿、面向一线需求、面向行业痛点”的发展方向，提供专业、可信的区块链安全服务。

实验室现有信息安全专业博士 2 名，高级安全工程师 5 名，拥有区块链风险评估平台、智能合约安全审计平台、DApp 安全监测平台、数字资产综合防护平台、量子随机数服务系统、私钥部分缺失找回系统、数字资产流转可视化系统等软硬件安全设施，涵盖比特币、以太坊、EOS、MOAC、井通等多款主流公链平台，可以提供七大类 28 种面向区块链产业的安全服务（详见附录）。

目前，实验室业务范围涵盖区块链安全技术研究、区块链安全测评、安全开发培训、智能合约安全审计、钱包和交易所的安全检测与加固、区块链安全咨询、区块链应用安全动态监测与通报、安全事件应急响应等多个领域，致力于发展成为“国际一流、国内领先”的区块链安全专业实验室。

更多信息，请访问实验室网站：www.mozi.one

安全服务：service@mozi.one

QQ/微信：12555615

微信公众号：mozilab

墨子安全实验室安全服务体系			
序号	类别	服务名称	服务描述
1	安全审计类	区块链源码安全审计	针对区块链底层系统（如BitCoin、ETH、EOS、MOAC、Frabic等）的源代码的安全问题进行分析和检测并验证
2		Dapp/智能合约源码安全审计	针对智能合约或分布式应用（DAPP）的核心源代码的安全问题进行分析和检测并验证
3		钱包代码安全审计	针对区块链钱包的核心源代码的安全问题进行分析和检测并验证
4		交易所代码安全审计	针对交易所代码进行安全扫描和审计，发现潜在安全问题和漏洞，并提出修复建议
5		矿池安全审计	可以受矿池运营方或者矿工团队的委托，对矿池进行全方位安全审计，包括矿池系统、费用分配等
6	安全评估类	区块链底层安全检测与风险评估	基于常见的安全漏洞和安全问题，对区块链底层进行渗透测试和漏洞挖掘，发现潜在的安全问题，并提出安全修复建议。
7		Dapp/智能合约安全检测与风险评估	基于常见的安全漏洞和安全问题，对Dapp/智能合约进行渗透测试和漏洞挖掘，发现潜在的安全问题，并提出安全修复建议
8		交易所安全检测、风险评估和安全加固	基于交易所的业务流程、应用场景和管理制度，进行安全检测和风险评估，并针对发现的问题提出安全加固建议
9		钱包安全检测与风险评估	基于钱包模块进行安全检测和漏洞挖掘，发现潜在安全问题和漏洞，并提出修复建议
10	安全保障类	区块链底层系统安全保障服务	针对区块链底层系统提供三类服务（建议按季度定期开展） 1）安全扫描：对漏洞、缺陷等开展静态扫描； 2）安全检测：在静态扫描基础上，增加模拟用户行为的检测。 3）安全加固：针对扫描和检测结果提供安全加固建议。
11		智能合约/DAPP安全保障服务	针对智能合约/DAPP提供三类服务（建议按月定期开展） 1）安全扫描：对漏洞、缺陷等开展静态扫描； 2）安全检测：在静态扫描基础上，增加模拟用户行为的检测。 3）安全加固：针对扫描和检测结果提供安全加固建议。
12		区块链钱包安全保障服务	针对区块链钱包提供三类服务（建议按季度定期开展） 1）安全扫描：对漏洞、缺陷等开展静态扫描； 2）安全检测：在静态扫描基础上，增加模拟用户行为的检测。 3）安全加固：针对扫描和检测结果提供安全加固建议。
13		交易所安全保障服务	针对交易所提供三类服务（建议按季度定期开展） 1）安全扫描：对漏洞、缺陷等开展静态扫描； 2）安全检测：在静态扫描基础上，增加模拟用户行为的检测。 3）安全加固：针对扫描和检测结果提供安全加固建议。
14		个人数字资产安全保障服务	针对持有数字资产的个人用户，提供三类服务： 1）安全扫描（事前）：对用户钱包运行环境做安全扫描； 2）安全使用（事中）：私钥安全托管和资产转移安全保障服务； 3）安全弥补（事后）：私钥部分缺失找回和丢币追踪服务

审计（签名）：周苏静 复核（签名）：苗知秋 签发（签名）：苗知秋

15	安全培训类	开发人员安全培训	针对区块链行业开发人员，进行安全培训，使其了解区块链常见安全问题和错误的开发习惯，学习如何进行安全开发
16		普通用户安全培训	面向普通用户，提供基本的安全意识培训和即学即用的安全建议，避免丢币、受骗等安全风险
17		管理层安全培训	面向政府和企业管理人员，培训内容主要包括：信息安全工作的当前形势、价值和必要性，近期发展态势等
18	开发支撑类	Dapp开发和安全审计	根据用户需求，设计和编写相应的智能合约和Dapp，在设计和编写过程中进行严格的代码审计和测试，特别适合较高安全性要求的区块链应用
19		已部署智能合约的整体迁移	已部署的智能合约，如果发现问题需要修改，需要做迁移操作。对于功能复杂的智能合约，迁移是一件非常精密和危险的操作。我们提供一整套的合约迁移方案，实现已部署和发币的智能合约的安全升级。
20		量子随机数服务	随机数是信息安全的基础。区块链本身无法产生真随机数，导致安全问题频发。我们提供基于量子器件的真随机数服务，安全度极高，访问接口包括web service、ETH合约、EOS合约和MOAC合约。
21	安全咨询类	区块链安全技术咨询	提供全面的区块链安全咨询服务，提供精准独到的专业见解。
22		区块链安全情报服务	提供专业及时的区块链安全情报服务，包括最新安全动态、安全事件和安全技术等。
23		基于区块链的信息系统安全方案设计	面向上链企事业用户，提供应用上链的技术方案设计，特别包含全面的安全方案设计和加固建议。
24		安全顾问服务	提供定制化安全服务，适合交易所等安全运营要求高的机构
25	安全应急类	部分私钥缺失找回	用户因为各种主观、客观原因丢失部分私钥，我们提供私钥找回服务。
26		丢币追踪服务	对于用户丢失的数字币的流转情况进行持续追踪，以便搜集尽可能多的信息，协助丢币找回
27		隐私保护服务	对用户的数字资产进行保护，避免被追踪/窥探，有效保障用户的隐私权
28		区块链安全事件响应	对安全事件进行应急响应和深入溯源分析，并提出安全修复建议；特别针对丢币问题进行应急响应和事件分析，并给出挽回建议；
备注	1. 如果预算不充足，可以选取重要功能及核心模块进行安全检测，其他非重要功能采用一般性测试（需另外收费）； 2. 如有尚未列入的服务需求，欢迎反馈至service@mozi.one 3. 更多详情可关注墨子安全实验室订阅号(mozilab)或访问墨子安全实验室网站（www.mozi.one）		

审计（签名）：周苏静 复核（签名）：苗知秋 签发（签名）：苗知秋