

```

1  pragma solidity ^0.4.0;
2
3  contract AccessControlMethod{
4
5      address public owner;
6      address public subject;
7      address public object;
8      Judge public jc;
9
10     event ReturnAccessResult(
11         address indexed _from,
12         string _errmsg,
13         bool _result,
14         uint _time,
15         uint _penalty
16     );
17
18     struct Misbehavior{
19         string res;                //resource on which the misbehavior is conducted
20         string action;            //action (e.g., "read", "write", "execute") of the misbehavior
21         string misbehavior;        //misbehavior
22         uint time;                //time of the misbehavior occurred
23         uint penalty;            //penalty opposed to the subject (number of minutes blocked)
24     }
25
26     struct BehaviorItem{          //for one resource
27         Misbehavior [] mbs;        //misbehavior list of the subject on a particular resource
28         uint TimeofUnblock;        //time when the resource is unblocked (0 if unblocked; otherwise,
blocked)
29     }
30
31     struct PolicyItem{           //for one (resource, action) pair;
32         bool isValued;            //for duplicate check
33         string permission;        //permission: "allow" or "deny"
34         uint minInterval;         //minimum allowable interval (in seconds) between two successive requests
35         uint ToLR;               //Time of Last Request
36         uint NoFR;               //Number of frequent Requests in a short period of time
37         uint threshold;          //threshold on NoFR, above which a misbehavior is suspected
38         bool result;             //last access result
39         uint8 err;               //last err code
40     }
41
42     mapping (bytes32 => mapping(bytes32 => PolicyItem)) policies; //mapping (resource, action) =>
PolicyCriteria for policy check
43     mapping (bytes32 => BehaviorItem) behaviors; //mapping resource => BehaviorCriteria for behavior
check
44
45     /*convert strings to byte32*/
46     function stringToBytes32(string _str) public constant returns (bytes32){
47         bytes memory tempBytes = bytes(_str);
48         bytes32 convertedBytes;
49         if(0 == tempBytes.length){
50             return 0x0;
51         }
52         assembly {
53             convertedBytes := mload(add(_str, 32))
54         }
55         return convertedBytes;
56     }
57
58     function AccessControlMethod(address _subject) public{
59         owner = msg.sender;
60         object = msg.sender;
61         subject = _subject;
62     }
63

```

```

64     function setJC(address _jc)public{
65         if(owner == msg.sender){
66             jc = Judge(_jc);
67         }
68         else throw;
69     }
70 }
71
72     function policyAdd(string _resource, string _action, string _permission, uint _minInterval, uint
_threshold) public{
73         bytes32 resource = stringToBytes32(_resource);
74         bytes32 action = stringToBytes32(_action);
75         if(msg.sender == owner){
76             if(policies[resource][action].isValued) throw; //duplicated key
77             else{
78                 policies[resource][action].permission = _permission;
79                 policies[resource][action].minInterval = _minInterval;
80                 policies[resource][action].threshold = _threshold;
81                 policies[resource][action].ToLR = 0;
82                 policies[resource][action].NoFR = 0;
83                 policies[resource][action].isValued = true;
84                 policies[resource][action].result = false;
85                 behaviors[resource].TimeofUnblock = 0;
86             }
87         }
88         else throw;
89     }
90
91     function getPolicy(string _resource, string _action) public constant returns (string _permission, uint
_minInterval, uint _threshold, uint _ToLR, uint _NoFR, bool _res, uint8 _errcode){
92         bytes32 resource = stringToBytes32(_resource);
93         bytes32 action = stringToBytes32(_action);
94         if(policies[resource][action].isValued){
95             _permission = policies[resource][action].permission;
96             _minInterval = policies[resource][action].minInterval;
97             _threshold = policies[resource][action].threshold;
98             _NoFR = policies[resource][action].NoFR;
99             _ToLR = policies[resource][action].ToLR;
100             _res = policies[resource][action].result;
101             _errcode = policies[resource][action].err;
102         }
103         else throw;
104     }
105 }
106
107     function policyUpdate(string _resource, string _action, string _newPermission) public{
108         bytes32 resource = stringToBytes32(_resource);
109         bytes32 action = stringToBytes32(_action);
110         if(policies[resource][action].isValued){
111             policies[resource][action].permission = _newPermission;
112         }
113         else throw;
114     }
115
116     function minIntervalUpdate(string _resource, string _action, uint _newMinInterval) public{
117         bytes32 resource = stringToBytes32(_resource);
118         bytes32 action = stringToBytes32(_action);
119         if(policies[resource][action].isValued){
120             policies[resource][action].minInterval= _newMinInterval;
121         }
122         else throw;
123     }
124
125     function thresholdUpdate(string _resource, string _action, uint _newThreshold) public{
126         bytes32 resource = stringToBytes32(_resource);
127         bytes32 action = stringToBytes32(_action);

```

```

128         if(policies[resource][action].isValued){
129             policies[resource][action].threshold= _newThreshold;
130         }
131         else throw;
132     }
133
134     function policyDelete(string _resource, string _action) public{
135         bytes32 resource = stringToBytes32(_resource);
136         bytes32 action = stringToBytes32(_action);
137         if(msg.sender == owner){
138             if(policies[resource][action].isValued){
139                 delete policies[resource][action];
140             }
141             else throw;
142         }
143         else throw;
144     }
145
146     /*Use event*/
147     function accessControl(string _resource, string _action, uint _time) public{
148
149         bool policycheck = false;
150         bool behaviorcheck = true;
151         uint8 errcode = 0;
152         uint penalty = 0;
153
154         if (msg.sender == subject){
155             bytes32 resource = stringToBytes32(_resource);
156             bytes32 action = stringToBytes32(_action);
157
158             if(behaviors[resource].TimeofUnblock >= _time){//still blocked state
159                 errcode = 1; //"Requests are blocked!"
160             }
161
162             else{//unblocked state
163                 if(behaviors[resource].TimeofUnblock > 0){
164                     behaviors[resource].TimeofUnblock = 0;
165                     policies[resource][action].NoFR = 0;
166                     policies[resource][action].ToLR = 0;
167                 }
168                 //policy check
169                 if (keccak256("allow") == keccak256(policies[resource][action].permission)){
170                     policycheck = true;
171                 }
172                 else{
173                     policycheck = false;
174                 }
175                 //behavior check
176                 if (_time - policies[resource][action].ToLR <= policies[resource][action].minInterval){
177                     policies[resource][action].NoFR++;
178                     if(policies[resource][action].NoFR >= policies[resource][action].threshold){
179                         penalty = jc.misbehaviorJudge(subject, object, _resource, _action, "Too frequent
access!", _time);
180                         behaviorcheck = false;
181                         behaviors[resource].TimeofUnblock = _time + penalty * 1 minutes;
182                         behaviors[resource].mbs.push(Misbehavior(_resource, _action, "Too frequent
access!", _time, penalty));//problem occurs when using array
183                     }
184                 }
185                 else{
186                     policies[resource][action].NoFR = 0;
187                 }
188                 if(!policycheck && behaviorcheck) errcode = 2; //"Static Check failed!"
189                 if(policycheck && !behaviorcheck) errcode = 3; //"Misbehavior detected!"
190                 if(!policycheck && !behaviorcheck) errcode = 4; //"Static check failed! & Misbehavior
detected!";

```

```

191         }
192         policies[resource][action].ToLR = _time;
193     }
194     else {
195         errcode = 5; //"Wrong object or subject detected!";
196     }
197     policies[resource][action].result = policycheck && behaviorcheck;
198     policies[resource][action].err = errcode;
199     if(0 == errcode) ReturnAccessResult(msg.sender, "Access authorized!", true, _time, penalty);
200     if(1 == errcode) ReturnAccessResult(msg.sender, "Requests are blocked!", false, _time, penalty);
201     if(2 == errcode) ReturnAccessResult(msg.sender, "Static Check failed!", false, _time, penalty);
202     if(3 == errcode) ReturnAccessResult(msg.sender, "Misbehavior detected!", false, _time, penalty);
203     if(4 == errcode) ReturnAccessResult(msg.sender, "Static check failed! & Misbehavior detected!",
false, _time, penalty);
204     if(5 == errcode) ReturnAccessResult(msg.sender, "Wrong object or subject specified!", false, _time,
penalty);
205     }
206
207
208     function getTimeofUnblock(string _resource) public constant returns(uint _penalty, uint
_timeOfUnblock){
209         bytes32 resource= stringToBytes32(_resource);
210         _timeOfUnblock = behaviors[resource].TimeofUnblock;
211         uint l = behaviors[resource].mbs.length;
212         _penalty = behaviors[resource].mbs[l - 1].penalty;
213     }
214
215
216     function deleteACC() public{
217         if(msg.sender == owner){
218             selfdestruct(this);
219         }
220     }
221 }
222
223
224 contract Judge{
225     function misbehaviorJudge(address _subject, address _object, string _res, string _action, string
_misbehavior, uint _time) public returns (uint );
226 }
227

```