```
1  contract DAO
2  {
3    mapping (address => uint) private userBalances;
4
5    function withdrawBalance () public {
6      uint amountTowithdraw = userBalances[msg.sender];
7      (bool success,) = msg.sender.call.value(
           amountTowithdraw)("");
8      require (success);
9      userBalances[msg.sender] = 0;
10   }
11 }
```

Listing 1: Problematic Contract

```
1   contract DAO
2   {
3     mapping (address => uint) private userBalances;
4     mapping (address => int[size_of_blocks]) private
          visitMap;
5     a graph object CFG;
6
7     function withdrawBalance () public {
8
9       if(!check(1,2))
10        Do something... (e.g., revert())
11      uint amountTowithdraw = userBalances[msg.sender];
12      (bool success,) = msg.sender.call.value(
            amountTowithdraw)("");
13      record(2);
14
15      if(!check(2,3))
16        Do something... (e.g., revert())
17      require (success);
18      userBalances[msg.sender] = 0;
19      record(3);
20    }
21
22    function check(cur_block, next_block) {
23      for each (block_no in CFG.next_block(cur_block))
24        if (next_block in CFG.next(cur_block))
25          return 1;
26        else return 0;
27    }
28
29    function record(cur_block) {
30      visitMap[tx.origin][cur_block] = 1;
31    }
32  }
33 }
```

Listing 2: Instrumented Contract