算法：

根据taint传播，标记Call Graph中节点是否为taint

1：定位到含有转账功能的function以及相应的节点 ---分别记为function\_transEth, node\_eth

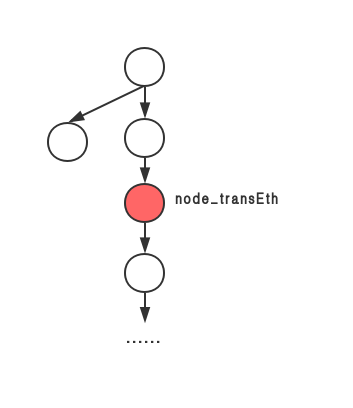


图 1 function\_transEth的CFG

2：以node\_transEth为起始节点往后遍历cfg，以便寻找taint\_node，若找到直接输出function\_transEth为reentrancy,若没找到，则转第3步

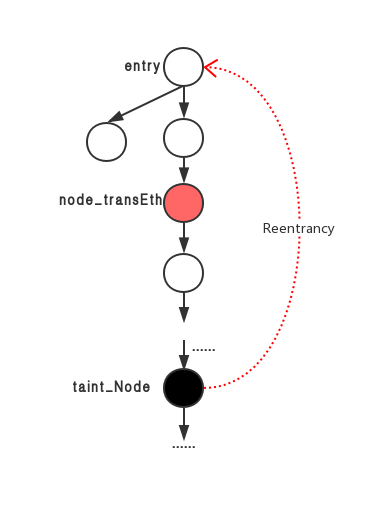


图 2function\_transEth还未找其callee就已经满足Reentrancy

3：以第一步的function\_transEth为起点，在call graph中寻找所有标记为taint的函数节点---记为taint\_function\_callee

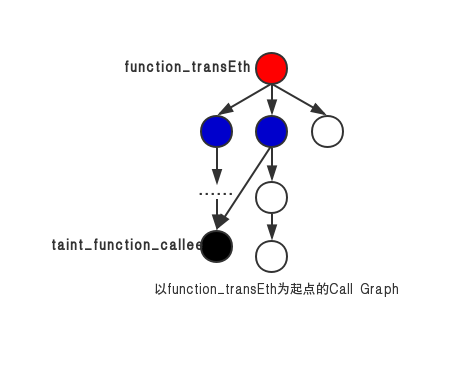
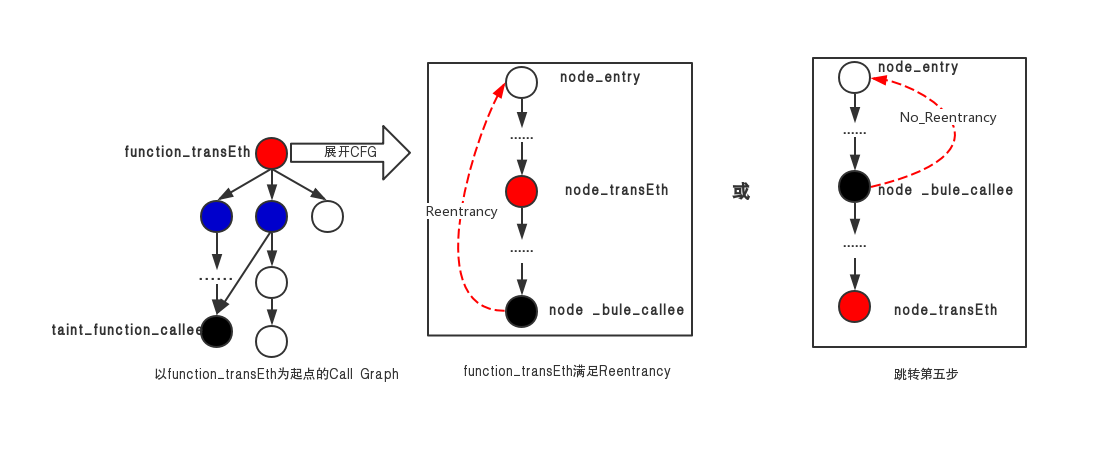


图 3Call Graph\_function\_transEth为起点

4：遍历function\_eth的cfg，判断node to taint\_callee是否在node\_eth之后。 若是则输出function\_transEth为reentrancy， 若不是则转第5步



5：在call graph找到function\_transEth的taint\_funtion\_caller。

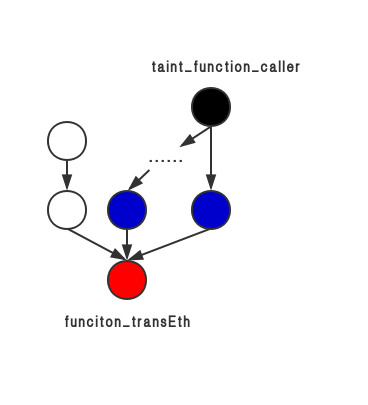
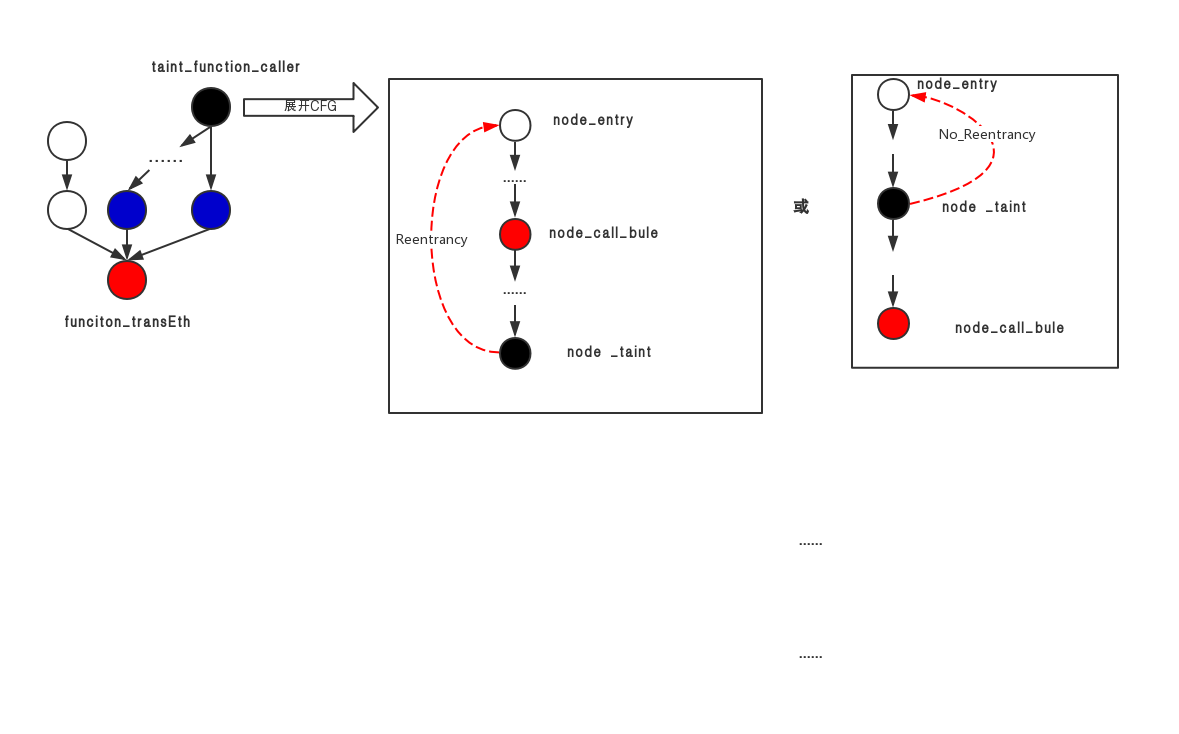


图 4Call Graph

6：遍历taint\_caller的cfg判断是否满足Reentrancy结构



7：根据以上6步可以得到candidate path of reentrancy。选出最短path 即为所求