带解决小点：

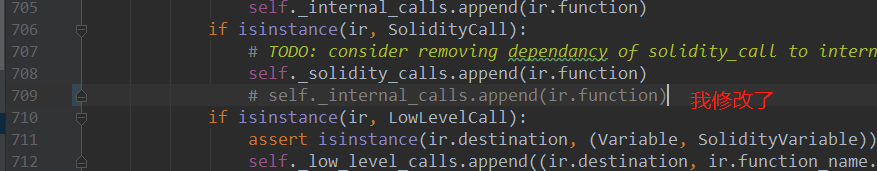
eth = =0.应该判断为不可以reentrancy

for node in function.nodes:

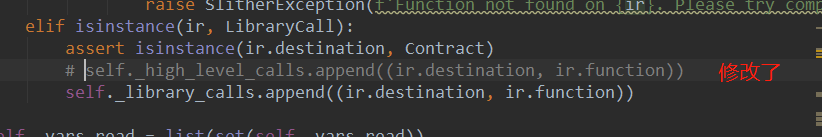
for ir in node.irs\_ssa:

2017年7月27

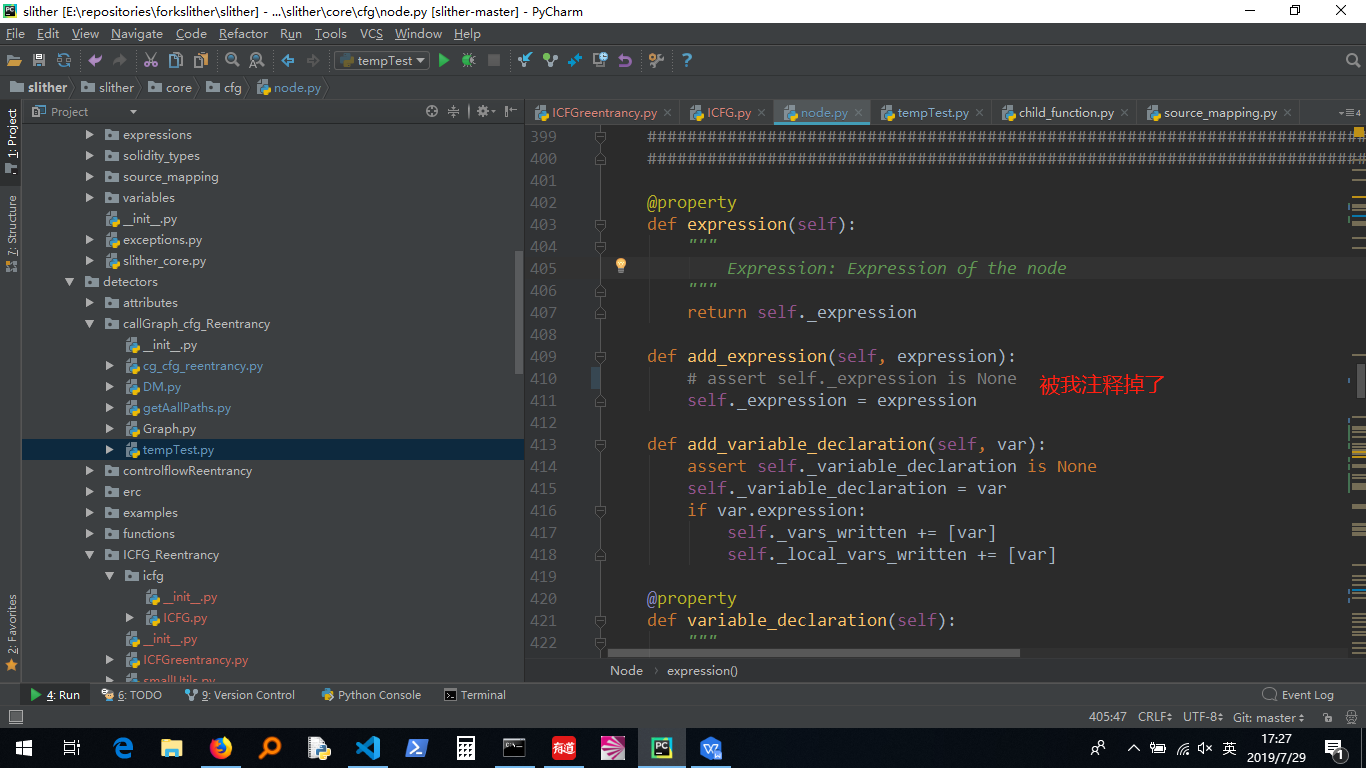
E:\repositories\forkslither\slither\slither\core\cfg\node.py



E:\repositories\forkslither\slither\slither\core\cfg\node.py



E:\repositories\forkslither\slither\slither\core\cfg\node.py



test3662.sol

INFO:Slither:./src\_contracts\_2\_3/test1688.sol analyzed (17540 contracts), 0 result(s) found

**检测的数据集量17540 contracts 输出统计结果如下 单位：函数**

|  |  |  |
| --- | --- | --- |
|  | detector 加入DM（目前还是rule策略） | detector 中不加入DM |
| [ ICFG ] | 117 | 136 |
| [ CFG ] | 135 | 334 |

SlitherIR:

addr = msg.sender;

addr(address) := msg.sender(address)

addr.call();

TMP\_10(bool) = LOW\_LEVEL\_CALL, dest:addr, function:call, arguments:[]

erc20 = ERC20(\_addr)

TMP\_9 = CONVERT \_addr to ERC20

erc20(ERC20) := TMP\_9(ERC20)

erc20 = Victim(this).returnERC20(\_addr);

TMP\_9 = CONVERT this to Victim

TMP\_11(ERC20) = HIGH\_LEVEL\_CALL, dest:TMP\_9(Victim), function:returnERC20, arguments:['\_addr']

erc20(ERC20) := TMP\_11(ERC20)

erc20 = returnERC20(\_addr)

TMP\_9(ERC20) = INTERNAL\_CALL, Victim.returnERC20(address)(\_addr)

erc20(ERC20) := TMP\_9(ERC20)

erc20 = ERC20(players[0])

REF\_4(address) -> players[0]

TMP\_9 = CONVERT REF\_4 to ERC20

erc20(ERC20) := TMP\_9(ERC20)

players[0] = \_address

REF\_0(address) -> players[0]

REF\_0 (->players) := \_address(address)

**Taint 传播表**

|  |  |
| --- | --- |
| **source code** | **IR** |
| addr = msg.sender; 源头直接赋值 | addr(address) := msg.sender(address) |
| erc20 = ERC20(\_addr) 源头地址绑定 | TMP\_9 = CONVERT \_addr to ERC20  erc20(ERC20) := TMP\_9(ERC20) |
| erc20 = Victim(this).returnERC20(\_addr);  源头作为参数，传递给合约外部函数returnERC20  然后该函数返回值又进行taint传播 | TMP\_9 = CONVERT this to Victim  TMP\_11(ERC20)=HIGH\_LEVEL\_CALL,dest:TMP\_9(Victim), function:returnERC20, arguments:['\_addr']  erc20(ERC20) := TMP\_11(ERC20) |
| erc20 = returnERC20(\_addr)  源头作为参数，传递给合约内部函数returnERC20  然后该函数返回值又进行taint传播 | TMP\_9(ERC20) = INTERNAL\_CALL, Victim.returnERC20(address)(\_addr)  erc20(ERC20) := TMP\_9(ERC20) |
| erc20 = ERC20(players[0])  players是个address[], 但players被taint源头赋值 | REF\_4(address) -> players[0]  TMP\_9 = CONVERT REF\_4 to ERC20  erc20(ERC20) := TMP\_9(ERC20) |

**去掉与合约和函数都重复的情况下**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PPT1 | PPT2 | PPT3 | PPT4 | PPT5 | other |
| 青区 | 0 | 14 | 8 | 0 | 0 | 0 |
| 白区 | 0 | 0 | 0 | 0 | 0 | 0 |

**数据解释：**

**青区：**

PPT1：PPT1与PPT3完成的防御原理是一样的，故很多程序员会选择PPT3，而不是PPT1，故PPT1少

PPT4：lock本来就少

PPT5: slither就应该是0，事实也就是0

**白区：**

本来白区就是3交重地，所以TP较多，且FP都没有落在1~5000.故为0

**ps:**

此时数据不好看没关系，蓝区才是大头。