**ppt5全局变量在支付之前更新**

**out3671.sol（securify ppt5类型FP） (8月15号)**

contract PvPCrash {

function withdraw() gasMin isHuman public returns (bool) {

address \_user = msg.sender;

uint256 \_userBalance;

if (!roundEnded && withdrawBlock[block.number] <= maxNumBlock) {

\_userBalance = getBalance(\_user);

if (\_balance > \_userBalance) {

if (\_userBalance > 0) {

**\_user.transfer(\_userBalance); // cfg的后面只有return true了**

emit Withdraw(\_user, \_userBalance);

}

return true;

}

}

return true;

}

}

**out7854.sol(oyente PPT5类型FP) （8月15号）**

contract MultiSeller{

function sell( IMultiToken \_mtkn, uint256 \_amount, bytes \_callDatas, uint[] \_starts, address \_for) public

{

\_mtkn.asmTransferFrom(msg.sender, this, \_amount);

\_mtkn.unbundle(this, \_amount);

change(\_callDatas, \_starts);

**\_for.transfer(address(this).balance);**

}

}

**out2363.sol**

**新的两个例子oyente FP**

contract Crowdsale{

function withdrawalOwner() public{

uint amount = **this.balance**;

if (**amount** > 0) {

bool ok = **msg.sender.call.value(amount)(); // amount第二次会变0**

WithdrawalEther(msg.sender, amount, ok);

}

}

}

**out7219.sol**

contract GreedVSFear {

function jackpotSend() payable public {

uint256 **ethToPay** = SafeMath.sub(jackpotCollected, jackpotReceived);

**require(ethToPay > 1);**  **// 对payment中的钱数做了require**

**jackpotReceived = SafeMath.add(jackpotReceived, ethToPay); //全局变量的写**

if(!jackpotAddress.call.value(**ethToPay**).gas(400000)()) { **// payment ethToPay在之前被更新**

jackpotReceived = SafeMath.sub(jackpotReceived, ethToPay);

}

}

}

**out11123.sol**

contract PausableInterface {

function executeCall ( address \_target, uint256 \_suppliedGas, bytes \_transactionBytecode) external

{

**require(underExecution == false); //全局变量判断**

**underExecution = true; // Avoid recursive calling 全局变量更新**

\_target.call.gas(\_suppliedGas).value(\_ethValue)(\_transactionBytecode); //payment

**underExecution = false; //全局变量再次更新**

}

}

**out3485.sol (slither和securify都是FP)**

contract PowerEtherHelper{

function withdrawPendingTransactions() public returns (bool) {

uint256 amount = playerFundsToWithdraw[msg.sender];

**playerFundsToWithdraw[msg.sender] = 0; //全局变量更新**

if (**msg.sender.call.value(amount)()**) { //支付

return true;

} else {

playerFundsToWithdraw[msg.sender] = amount; //全局变量的写

return false;

}

}

}

**ppt1支付之前有access control**

**out6286.sol (slither和securify都是FP)**

contract CozyTimeAuction {

function buyCozy(uint256 \_pepeId, uint256 \_cozyCandidate, bool \_candidateAsFather, address \_pepeReceiver) public payable {

require(address(pepeContract) == msg.sender); // access control

PepeAuction storage auction = auctions[\_pepeId];

uint256 price = calculateBid(\_pepeId);

uint256 totalFee = price \* fee / FEE\_DIVIDER;

...

uint256 price = calculateBid(\_pepeId);

...

//Send ETH to seller

auction.seller.transfer(price - totalFee); //转账

if (\_candidateAsFather) {

if (!pepeContract.cozyTime(auction.pepeId, \_cozyCandidate, \_pepeReceiver)) { **//外部调用**

revert();

}

}

delete auctions[\_pepeId]; //真实案例确实有这句话，我多删除了。。。这是全局变量的写

}

}

**ppt5**

test142.sol

function buy\_the\_tokens() public{

require(!bought\_tokens); //全局变量

require(this.balance >= min\_required\_amount);

bought\_tokens = true; //全局变量更新

contract\_eth\_value = this.balance;

require(sale.call.value(contract\_eth\_value)()); //支付

bought\_tokens = false;

}

**ppt1**

function \_\_callback(bytes32 id, string res) public {

require(msg.sender == oraclize\_cbAddress(), "Permission denied"); //access control

require(games[id].bet > 0, "Game not found");

bool result = parseInt(res) == 1;

uint win = games[id].option == result ? winSize(games[id].bet) : 0;

if(win > 0) {

....

}

if(ethergames != address(0)) {

ethergames.call.value(games[id].bet / 100).gas(45000)(); //支付

}

delete games[id];

}

out2040

contract Exchange {

....

....

function buyPerUnit(address \_asset, uint256 \_amount) external payable {

require(blacklist[msg.sender] == false);

ListAsset storage listing = listOfAssets[\_asset];

require(\_amount <= listing.amount);

uint totalPrice = \_amount.mul(listing.price);

require(msg.value == totalPrice);

ERC20\_Interface token = ERC20\_Interface(\_asset); //static taint

if(token.allowance(owner,address(this)) >= \_amount){

assert(token.transferFrom(owner,msg.sender, \_amount));

owner.transfer(totalPrice); // 支付

listing.amount= listing.amount.sub(\_amount);

}

}

}