Real Wolrd CTF 3rd Writeup | Easy Defi

原创

LiRiu 于 2021-03-24 16:28:42 发布 141 收藏

分类专栏： 胖达的区块链安全工坊 文章标签： 区块链 智能合约 CTF writeup defi

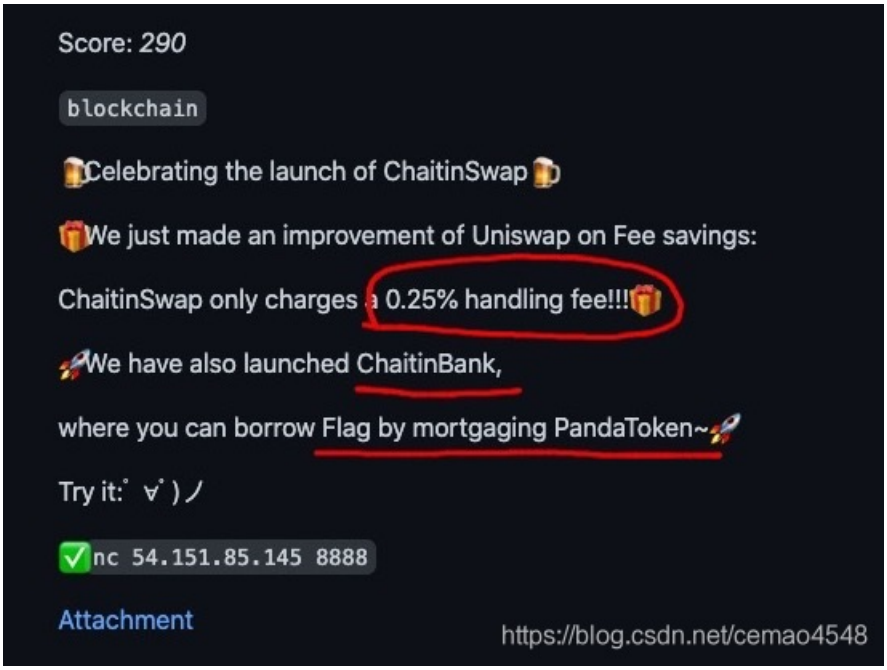
本文链接：https://blog.csdn.net/cemao4548/article/details/115180254

题目环境架构

EasyDefi是一道面向去中心化金融服务的智能合约题目。

题目代码通过附件的形式给出。

同时我们可以通过题目描述了解到
ChaitinSwap的逻辑和Uniswap基本类似，只是更改了手续费比例。



ChaitinBank听上去是某种提供了借贷服务的Defi项目，
我们看一下代码:

```
904
905       // Enter the bank. Pay some feifeicoin. Earn some Flags.
906       function depositFeiCointoFlag(uint256 _amount) public {
907           IERC20(feicoin).transferFrom(msg.sender, address(this), _amount);
908           uint256 valueOfFeicoin = calcFeiCoinValue(); // exchange feicoin to ChaitinCoin
909           uint256 exceptFlagsAmount = valueOfFeicoin * ( _amount / 10 ) / CTTperFlag;
910
911           uint256 flagsAmount = 0;
912           if(exceptFlagsAmount <= flagToken.balanceOf(address(this))){
913               flagsAmount = exceptFlagsAmount;
914           }else{
915               flagsAmount = flagToken.balanceOf(address(this));
916           }
917
918           flagToken.transfer(msg.sender, flagsAmount);
919
920       }
921
922       // Leave the Flags. Claim back your FeiCoins.
923       function leaveFlagstoFeiCoin(uint256 _amount) public {  //TODO: how to define
924           flagToken.transferFrom(msg.sender, address(this), _amount);
925           uint256 valueOfFeicoin = calcFeiCoinValue();
926           uint256 exceptFeicoinAmount = _amount * CTTperFlag / valueOfFeicoin;
927
928           uint256 feicoinsAmount = 0;
929           if (exceptFeicoinAmount <= IERC20(feicoin).balanceOf(address(this))){
930               feicoinsAmount = exceptFeicoinAmount;
931           }else{
932               feicoinsAmount = IERC20(feicoin).balanceOf(address(this));
933           }
934
935           IERC20(feicoin).transfer(msg.sender, feicoinsAmount);
936       }
937   }
```


Bank仅提供了两个功能接口，一个用FeiCoin换Flag，另一个使用Flag换FeiCoin。
但是这个FeiCoin又是个啥？我们到ERC20.sol里面确认一下：

```
239   contract FlagToken is ERC20("Flag", "FLAG"), Ownable {
240       function mint(address _to, uint256 _amount) public onlyOwner {
241           _mint(_to, _amount);
242       }
243       function burn(address _from ,uint _amount) public {
244           _burn(_from,_amount);
245       }
246   }
247
248   contract ChaitinToken is ERC20("ChangtingToken", "CTT"), Ownable {
249       function mint(address _to, uint256 _amount) public onlyOwner {
250           _mint(_to, _amount);
251       }
252       function burn(address _from ,uint _amount) public {
253           _burn(_from,_amount);
254       }
255   }
256
257   contract FeiToken is ERC20("PandaToken", "PDT"), Ownable {
258       function mint(address _to, uint256 _amount) public onlyOwner {
259           _mint(_to, _amount);
260       }
261       function burn(address _from ,uint _amount) public {
262           _burn(_from,_amount);
263       }
264   }
```


可见这里Panda币的合约名称叫FeiToken，看来FeiCoin和PandaToken是一个东西:

至此我们可以得到题目大概的组织框架如下:

一共存在三种代币: 长亭币、Panda币和FLAG
在ChaitinSwap中，长亭币 和 Panda币可以互换
在ChaitinBank中，Panda币和FLAG可以互换

代码审计

vuln1 K值不守恒

我们来审计合约代码，根据题目描述中的提示。
chaitinSwap和uniswap的区别主要在手续费。
chaitinSwap仅收取0.25%的手续费。
所以我们来看定义手续费附近的代码

```
737     // given an output amount of an asset and pair reserves, returns a required input amount of the other asset
738     function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) internal pure returns (uint amountIn) {
739         require(amountOut > 0, 'ChaitinLibrary: INSUFFICIENT_OUTPUT_AMOUNT');
740         require(reserveIn > 0 && reserveOut > 0, 'ChaitinLibrary: INSUFFICIENT_LIQUIDITY');
741         uint numerator = reserveIn.mul(amountOut).mul(10000);
742         uint denominator = reserveOut.sub(amountOut).mul(9975);
743         amountIn = (numerator / denominator).add(1);
744     }
745
```

这里getAmountIn/Out函数中，有对0.25%的描述。
除此之外，在Factory合约的swap中也有相关描述。

```
354     function swap(uint amount0Out, uint amount1Out, address to, bytes calldata data) external lock {
355         require(amount0Out > 0 || amount1Out > 0, 'Chaitin: INSUFFICIENT_OUTPUT_AMOUNT');
356         (uint112 _reserve0, uint112 _reserve1,) = getReserves(); // gas savings
357         require(amount0Out < _reserve0 && amount1Out < _reserve1, 'Chaitin: INSUFFICIENT_LIQUIDITY');
358
359         uint balance0;
360         uint balance1;
361         { // scope for _token{0,1}, avoids stack too deep errors
362         address _token0 = token0;
363         address _token1 = token1;
364         require(to != _token0 && to != _token1, 'Chaitin: INVALID_TO');
365         if (amount0Out > 0) _safeTransfer(_token0, to, amount0Out); // optimistically transfer tokens
366         if (amount1Out > 0) _safeTransfer(_token1, to, amount1Out); // optimistically transfer tokens
367         if (data.length > 0) IChaitinCallee(to).ChaitinCall(msg.sender, amount0Out, amount1Out, data);
368         balance0 = IERC20(_token0).balanceOf(address(this));
369         balance1 = IERC20(_token1).balanceOf(address(this));
370         }
371         uint amount0In = balance0 > _reserve0 - amount0Out ? balance0 - (_reserve0 - amount0Out) : 0;
372         uint amount1In = balance1 > _reserve1 - amount1Out ? balance1 - (_reserve1 - amount1Out) : 0;
373         require(amount0In > 0 || amount1In > 0, 'Chaitin: INSUFFICIENT_INPUT_AMOUNT');
374         { // scope for reserve{0,1}Adjusted, avoids stack too deep errors
375         uint balance0Adjusted = balance0.mul(10000).sub(amount0In.mul(25));
376         uint balance1Adjusted = balance1.mul(10000).sub(amount1In.mul(25));
377         require(balance0Adjusted.mul(balance1Adjusted) >= uint(_reserve0).mul(_reserve1).mul(1000**2), 'Chaitin: K');
378         }
379
380         _update(balance0, balance1, _reserve0, _reserve1);
381         emit Swap(msg.sender, amount0In, amount1In, amount0Out, amount1Out, to);
382     }
383
```

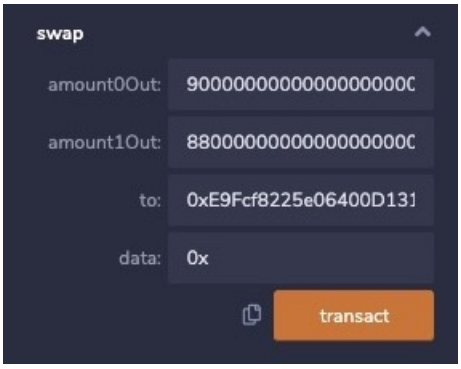
于是我们发现了第一个漏洞点，这个ChaitinSwap的K值 不守恒。
为了表示0.25%的手续费，除数从1000改成了10000。
但是在377行进行比较时， 还是和Uniswap代码中原本的`1000**2`进行比较。
这导致最后的这个K值守恒的检查，很容易通过。
Uniswap的K值守恒是用大于等于号检查的，这个比较神奇，可以阅读这篇文章来详细分析一下。`https://zhuanlan.zhihu.com/p/272241656`

这里我们需要做的只是让amount1In或者amount0In为正数。我们稍稍向合约中转账(比如题目给我们准备的1个长亭币)就可以满足此限制。
之后我们将取出的amount0Out和amount1Out写得大一些，只要balance0Ajusted和balance1Adjusted相乘不要过小即可。

这样我们就可以用1个长亭币换出了大量的PandaToken和长亭币。

具体操作如下，
第一步把1 个长亭币转到chaitinToken-feiToken pair 合约中

第二步 调用pair合约的swap函数，把钱从流通池里掏出来



至此， 我们成功从交易所中 盗取了90个PandaToken和 88个长亭币

vuln2 Bank没有用预言机，可以操纵币价

然而题目要求我们搞到80个Flag币

我们知道能得到FLAG的地方只有，BANK合约

```
905     // Enter the bank. Pay some feifeicoin. Earn some Flags.
906     function depositFeiCointoFlag(uint256 _amount) public {
907         IERC20(feicoin).transferFrom(msg.sender, address(this), _amount);
908         uint256 valueOfFeicoin = calcFeiCoinValue(); // exchange feicoin to ChaitinCoin
909         uint256 exceptFlagsAmount = valueOfFeicoin * ( _amount / 10 ) / CTTperFlag;
910
911         uint256 flagsAmount = 0;
912         if(exceptFlagsAmount <= flagToken.balanceOf(address(this))){
913             flagsAmount = exceptFlagsAmount;
914         }else{
915             flagsAmount = flagToken.balanceOf(address(this));
916         }
917
918         flagToken.transfer(msg.sender, flagsAmount);
919
920     }
921
```


而且我们可以看到，用户能得到多少个flag币，和存入的PandaToken的币价有关系

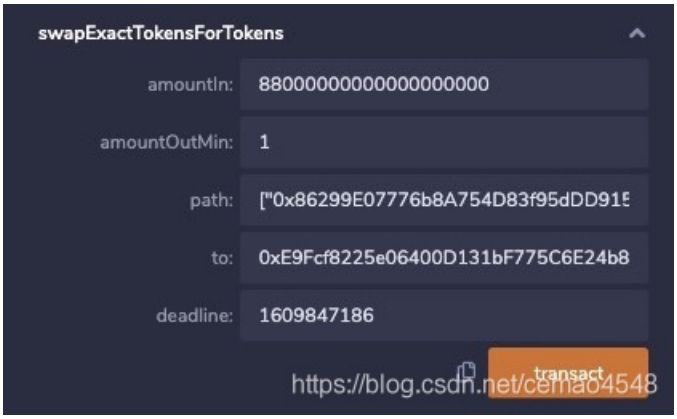
```
893     function calcFeiCoinValue() public view returns(uint256 value){
894         (uint256 reserve0, uint256 reserve1, uint32 time ) = IPokePair(chaitinFeifeiTokenPair).getReserves();
895         address token0 = IPokePair(chaitinFeifeiTokenPair).token0();
896         if(token0 == feicoin){
897             value = IPokeRouter01(chaitinSwap).getAmountOut( 10, reserve0, reserve1);
898         }else{
899             value = IPokeRouter01(chaitinSwap).getAmountOut( 10, reserve1, reserve0);
900         }
901
902         return value;
903     }
```

所以PandaToken价格越高，我们能得到的Flag越多。
此时的PandaToken币价为 11

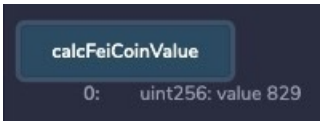
阅读代码我们发现，这里Bank合约没有使用预言机来防止恶意操纵币价。
所以很明显这里便是第二个漏洞点。
攻击者可以通过恶意操纵市场，来抬高PandaToken的币价。
我们通过Router合约可以实现该目的。

具体操作如下:
这里先Approve给Router全部的长亭币

然后调用Router.swapExactTokensForTokens 函数来拉动币价。
我们使用长亭币疯狂购买PandaToken，从而抬高PandaToken的币价。

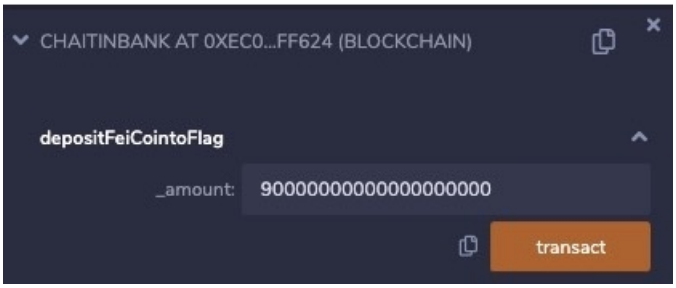


接着我们来到Bank合约，查看PandaToken的价格



发现PandaToken的币价已经高达829了

看到这么高的币价， 我们直接
调用depositFeiCointoFlag函数， 用手中全部的PandaToken来换取Flag

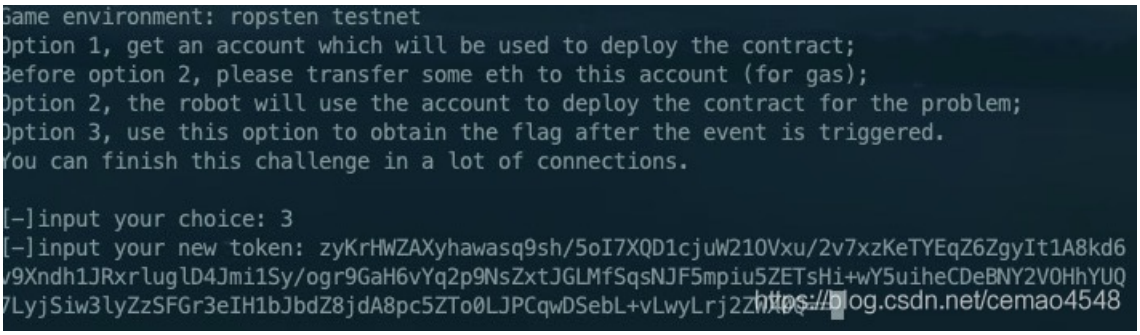


接下来，我们看看自己有多少个FlagToken

82.9个FLAG，满足题目要求。

获取FLAG

最后我们回到命令行界面，选择option3 并输入token



得到FLAG

`rwctf{axelytxtdznhahsebhk}`

相关链接

- 题目介绍和附件 https://github.com/chaitin/Real-World-CTF-3rd-Challenge-Attachments/tree/main/Easy%20Defi
- 为什么说Uniswap的K值不那么守恒 https://zhuanlan.zhihu.com/p/272241656
- Uniswap文档 https://uniswap.org/docs/v2/
- AMM自动做市商原理 https://www.jianshu.com/p/5bb44a9cd220