SharkTeam

Price Manipulation: Analysis of MonoX Finance's Hacked Event

Dec 02th, 2021

On November 30th, Beijing time, the liquidity pools of MonoX Finance on Ethereum and polygon were attacked, resulting in a loss of about $34 million, including about 2.1K WETH, 1.9M WMATIC, 36.1 WBTC, 143.4K MONO, 8.2 M USDC, 9.1M USDT, 1.2K LINK, 3.1K GHST, 5.1M DUCK, 4.1K MIM and 274.9 IMX.

SharkTeam conducted an attack analysis and technical analysis on this incident at the first time, and summarized the security precautions. It is hoped that the follow-up blockchain projects can learn from it and build a security line of defense in the blockchain industry.

1.Event analysis

Attacker Address 1:

0xecbe385f78041895c311070f344b55bfaa953258

Attacker address 2:

0x8f6a86f3ab015f4d03ddb13abb02710e6d7ab31b

Attack Contract 1:

0xf079d7911c13369e7fd85607970036d2883afcfd

Attack Contract 2:

0x119914de3ae03256fd58b66cd6b8c6a12c70cfb2

Attack Transaction 1:

https://etherscan.io/tx/0x9f14d093a2349de08f02fc0fb018dadb449351d0cdb7d0738ff69cc6fef5f299

Attack Transaction 2:

https://polygonscan.com/tx/0x5a03b9c03eedcb9ec6e70c6841eaa4976a732d050a6218969e39483bb3004d5d

Take transaction 1 on ETH as an example:

(1) First, the attacker calls Monoswap.swapExactTokenForToken to exchange 79.986094311542621010 MONO with 0.1 WETH as the attack principal.

(2) With the liquidity of 0x7b9aa6 removed, 1670.7572297649224 MONO and 6.862171986812230290 vCASH were removed.

(3) To remove the liquidity of cowrie.eth, 152.9745213857155 MONO and 0.628300423692773565 vCASH were removed.

(4) The liquidity of 0xab5167 was removed, 99940.7413658327 MONO and 410.478879590637971405 vCASH were removed.

The purpose of removing the liquidity of other users is to reduce the liquidity of MONO in the Monoswap contract to a smaller value, which is convenient for subsequent use of less MONO for price manipulation.。

The attacker can only remove the liquidity of other liquidity providers.

The reason is that the _removeLiquidityHelper, removeLiquidity and _removeLiquidity functions involved in removing liquidity in the contract do not authenticate the caller and to address, and the attacker can remove the liquidity. In addition to the liquidity of any user in the pool.

```
// actually removes liquidity
function removeLiquidity (address _token, uint256 liquidity, address to,
  uint256 minVcashOut,
  uint256 minTokenOut) external returns(uint256 vcashOut, uint256 tokenOut)  {
  (vcashOut, tokenOut) = _removeLiquidityHelper (_token, liquidity, to, minVcashOut, minT
}

// actually removes liquidity
function _removeLiquidityHelper (address _token, uint256 liquidity, address to,
  uint256 minVcashOut,
  uint256 minTokenOut,
  bool isETH) lockToken(_token) internal returns(uint256 vcashOut, uint256 tokenOut)  {
  require (tokenPoolStatus[_token]==1, "MonoX:NO_TOKEN");
  PoolInfo memory pool = pools[_token];
  uint256 poolValue;
  uint256 liquidityIn;
  (poolValue, liquidityIn, vcashOut, tokenOut) = _removeLiquidity(_token, liquidity, to);
  _mintFee(pool.pid, pool.lastPoolValue, poolValue);
  require (vcashOut>=minVcashOut, "MonoX:INSUFF_vCash");
  require (tokenOut>=minTokenOut, "MonoX:INSUFF_TOKEN");
```

(5) The attacker added liquidity to MONO and performed 55 MONO-MONO Swap to manipulate the price of MONO tokens.

The price of the Swap token in the Monoswap.sol contract through the getAmountOut function.

```
(tokenInPrice, tokenOutPrice, amountOut, tradeVcashValue) = getAmountOut(tokenIn, tokenOut, amountIn);
```

The input/input token price calculation logic in the getAmountOut function is as follows.

```
    tokenInPrice = _getNewPrice(tokenInPoolPrice, tokenInPoolTokenBalance,
      amountInWithFee, 0, TxType.SELL);
    tradeVcashValue = _getAvgPrice(tokenInPoolPrice, tokenInPrice).mul(amountInWithFee)/1e18;
  }

  if(tokenOut==vcashAddress){
    amountOut = tradeVcashValue;
    tokenOutPrice = 1e18;
  }else{
    require (tokenPoolStatus[tokenOut]==1, "MonoX:NO_POOL");
    // PoolInfo memory tokenOutPool = pools[tokenOut];
    PoolStatus tokenOutPoolStatus = pools[tokenOut].status;
    uint tokenOutPoolPrice = pools[tokenOut].price;
    uint tokenOutPoolTokenBalance = pools[tokenOut].tokenBalance;

    require (tokenOutPoolStatus != PoolStatus.UNLISTED, "MonoX:POOL_UNLST");

    amountOut = tradeVcashValue.add(tokenOutPoolTokenBalance.mul(tokenOutPoolPrice).div(1e18));
    amountOut = tradeVcashValue.mul(tokenOutPoolTokenBalance).div(amountOut);

    bool allowDirectSwap=directSwapAllowed(tokenInPoolPrice,tokenOutPoolPrice,tokenInPoolTokenBalance,tokenOutPoolTokenBal

    // assuming p1*p2 = k, equivalent to uniswap's x * y = k
    uint directSwapTokenOutPrice = allowDirectSwap?tokenInPoolPrice.mul(tokenOutPoolPrice).div(tokenInPrice):uint(-1);

    // prevent the attack where user can use a small pool to update price in a much larger pool
    tokenOutPrice = _getNewPrice(tokenOutPoolPrice, tokenOutPoolTokenBalance,
      amountOut, 0, TxType.BUY);
    tokenOutPrice = directSwapTokenOutPrice < tokenOutPrice?directSwapTokenOutPrice:tokenOutPrice;

    amountOut = tradeVcashValue.mul(1e18).div(_getAvgPrice(tokenOutPoolPrice, tokenOutPrice));
  }
```

```
function _getNewPrice (uint256 originalPrice, uint256 reserve,
  uint256 delta, uint256 deltaBlocks, TxType txType) pure internal returns(uint256 price) {
  if(txType==TxType.SELL) {
    // no risk of being div by 0
    price = originalPrice.mul(reserve)/(reserve.add(delta));
  }else{ // BUY
    price = originalPrice.mul(reserve).div(reserve.sub(delta));
  }
}
```

When incoming and outgoing are the same token MONO, tokenOutPrice > tokenInPrice. Since both tokenIn and tokenOut are MONO, the swapIn function will call _updateTokenInfo to update the price of MONO after calculating the price, which will cause the price of MONO to rise. The price of MONO was pulled to 843,741,636,512.366 vCASH/MONO.

(6) The attacker calls the swapTokenForExactToken function to exchange other tokens in the pool with MONO after the price has been raised.

Summary: The root cause of this attack is that the contract does not check the identity of the incoming and outgoing tokens of Swap, so that the attacker can use this "identity" vulnerability to control the price of the token, and use the token to raise the price of the token. Swap out all other tokens in the liquidity pool.

2. safety recommendations

This incident is another security incident related to lightning loans. The business logic design of the DeFi project is complex, involving many economic calculations and parameters, but the composability between different projects and agreements is extremely rich and difficult to predict. Prone to security breaches. As shown in the table below, the number of DeFi incidents that used flash loans to attack this new product has been endless in the past year, which has increased to 46. Generally, lightning loan attacks include four types: "raising arbitrage," "oracles manipulation," "re-entry attacks," and "technical vulnerabilities." This is a lightning loan attack based on the use of "oracles".

编号	日期	主攻协议	闪电贷协议	攻击类型
1	2020/2/15	bZx	dYdX	哄抬套利
2	2020/2/18	bZx	bZx	操纵预言机
3	2020/6/28	Balancer	dYdX	哄抬套利
4	2020/10/26	Harvest	Uniswap V2	操纵预言机
5	2020/11/6	Cheese Bank	dYdX	操纵预言机
6	2020/11/12	Akropolis	dYdX	重入攻击
7	2020/11/14	Value.DeFi	Aave/Uniswap V2	操纵预言机
8	2020/11/17	OUSD	dYdX	重入攻击
9	2020/12/18	Warp Finance	Uniswap V2/dYdX	操纵预言机
10	2021/2/4	Yearn.Finance	dYdX/Aave	哄抬套利
11	2021/2/13	Alpha.Finance	Aave	技术漏洞
12	2021/5/2	Spartan	PancakeSwap	哄抬套利
13	2021/5/8	Value.DeFi	Value.DeFi	技术漏洞
14	2021/5/8	Rari Capital	dYdX	重入攻击
15	2021/5/13	xToken	dYdX	操纵预言机
16	2021/5/16	bEarn Fi	Cream.Finance	技术漏洞
17	2021/5/20	PancakeBunny	PancakeSwap	操纵预言机
18	2021/5/22	Bogged Finance	PancakeSwap	技术漏洞
19	2021/5/25	AutoShark	PancakeSwap	技术漏洞+操纵预言机
20	2021/5/28	BurgerSwap	PancakeSwap	重入攻击
21	2021/5/28	JulSwap	JulSwap	操纵预言机
22	2021/5/30	Belt Finance	PancakeSwap	操纵预言机
23	2021/6/5	BurgerSwap	PancakeSwap	重入攻击
24	2021/6/10	EvoDeFi	PancakeSwap	技术漏洞
25	2021/6/19	Impossible Finance	PancakeSwap	技术漏洞
26	2021/6/23	Eleven Finance	ApeSwap	技术漏洞
27	2021/6/25	xWin Finance	PancakeSwap	操纵预言机
28	2021/7/2	XDXSwap	XDXSwap（Heco）	技术漏洞
29	2021/7/13	DeFiPie	PancakeSwap	重入攻击
30	2021/7/14	ApeRocket	PancakeSwap	技术漏洞
31	2021/7/14	ApeRocket	AAVE（Polygon）	技术漏洞
32	2021/7/16	Swamp.Finance	PancakeSwap	技术漏洞
33	2021/7/17	PancakeBunny	AAVE（Polygon）	技术漏洞
34	2021/7/18	Array.Finace	AAVE	技术漏洞
35	2021/8/4	Popsicle Finance	AAVE	技术漏洞
36	2021/8/4	Wault.Finance	PancakeSwap	技术漏洞
37	2021/8/17	XSURGE	PancakeSwap	重入攻击
38	2021/8/25	Dot.Finance	PancakeSwap	技术漏洞
39	2021/8/30	Cream Finance	Uniswap V2	重入攻击
40	2021/9/12	Zabu Finance	Pangolin	技术漏洞
41	2021/9/15	NowSwap	NowSwap	技术漏洞
42	2021/10/2	Twindex	Twindex	技术漏洞
43	2021/10/2	AutoShark	PancakeSwap	操纵预言机
44	2021/10/6	MY FARM PET	PancakeSwap	操纵预言机
45	2021/10/15	Indexed Finance	SushiSwap	技术漏洞
46	2021/10/20	PankeHunny	PancakeSwap	操纵预言机

SharkTeam reminds you to be vigilant when getting involved in blockchain projects, choose more stable, safer, and complete multiple rounds of auditing public chains and projects. You must not put your assets at risk and become hackers. Cash machine. As the project party, the security of smart contracts is vital to the safety of users' property! The blockchain project party should cooperate with a professional security audit company to conduct multiple rounds of audits to avoid status and calculation errors in the contract, and

provide guarantees for the security of users' digital assets and the security of the project itself.

As a leading blockchain security service team, SharkTeam provides developers with smart contract audit services. The smart contract audit service consists of manual audit and automated audit to meet the needs of different customers. It exclusively implements nearly two hundred audit contents covering the four aspects of high-level language layer, virtual machine layer, blockchain layer, and business logic layer, fully guaranteeing smart contracts Safety.


SharkTeam
团队介绍
SharkTeam专注于智能合约安全，由拥有多年一线网络安全实战经验的团队成员组成，精通区块链和智能合约底层原理，具备完善的区块链漏洞挖掘和智能合约审计能力，可提供全面的欺诈检测、威胁建模、合约审计、应急响应服务，已帮助多个知名区块链项目发现并修复安全漏洞，致力于保护用户数字资产安全与隐私安全。
安全漏洞库
虚拟机层
业务逻辑层
近200
独家实现
全覆盖
高级语言层
区块链层
安全服务
7*24小时在线
自动化审计
自动化扫描
全面覆盖
高精准度
人工审计服务
VIP安全审计服务
VIP合规审计服务
应急响应与合约定制
战略合作伙伴
OKEX
Huobi
Twitter：@sharkteamorg
Telegram：https://t.me/sharkteamorg

SharkTeam

In Math, We Trust !

https://t.me/sharkteamorg

https://twitter.com/sharkteamorg