

Beosin's Analysis of Team Finance's \$13M Exploit



Beosin's Analysis of Team Finance's \$13M Exploit

On October 27, 2022, Beosin EagleEye reported that Team Finance on Ethereum was exploited for over \$13M. The hacker illegally migrated WTH, CAW, USDC, TSUKA tokens from V2 liquidity pool to V3 liquidity pool by exploiting the vulnerability of the migrate function in LockToken contract, disrupted the Initialize price of V3 liquidity pool via sqrtPriceX96 to obtain a large amount of refund arbitrage. Beosin security team analyzed the incident and the results are as follows.

Related information

Attack transaction

0xb2e3ea72d353da43a2ac9a8f1670fd16463ab370e563b9b5b26119b2601277ce

Attacker's address

0x161cebB807Ac181d5303A4cCec2FC580CC5899Fd

0xBa399a2580785A2dEd740F5e30EC89Fb3E617e6E

Attack contract

oxCFFo7C4e6aa9E2fEco4DAaF5f41d1b1of3adAdF4

Victim contract

oxE2fE530Co47f2d85298bo7D9333Co5737f1435fB

Attack Flow

1. The migrate function of the victim contract (LockToken contract) supports the user to migrate the specified Uniswap-V2 liquidity to Uniswap-V3 liquidity. A part of the tokens will be refunded to the user based on the price after migration. The call to the function require: lock ID, lock time, and withdrawable address.

```
1531     function migrate(  
1532         uint256 _id,  
1533         IV3Migrator.MigrateParams calldata params,  
1534         bool noLiquidity,  
1535         uint160 sqrtPriceX96,  
1536         bool _mintNFT  
1537     )  
1538     external  
1539     payable  
1540     whenNotPaused  
1541     nonReentrant  
1542     {  
1543         require(address(nonfungiblePositionManager) != address(0), "NFT manager not set");  
1544         require(address(v3Migrator) != address(0), "v3 migrator not set");  
1545         Items memory lockedERC20 = lockedToken[_id];  
1546         require(block.timestamp < lockedERC20.unlockTime, "Unlock time already reached");  
1547         require(_msgSender() == lockedERC20.withdrawalAddress, "Unauthorised sender");  
1548         require(!lockedERC20.withdrawn, "Already withdrawn");
```

```
1568         uint256 refundEth = address(this).balance - ethBalanceBefore;  
1569         (bool refundSuccess,) = _msgSender().call.value(refundEth)("");  
1570         require(refundSuccess, 'Refund ETH failed');  
1571  
1572         uint256 token0BalanceAfter = IERC20(params.token0).balanceOf(address(this));  
1573         uint256 refundToken0 = token0BalanceAfter - token0BalanceBefore;  
1574         if( refundToken0 > 0 ) {  
1575             require(IERC20(params.token0).transfer(_msgSender(), refundToken0));  
1576         }  
1577  
1578         uint256 token1BalanceAfter = IERC20(params.token1).balanceOf(address(this));  
1579         uint256 refundToken1 = token1BalanceAfter - token1BalanceBefore;  
1580         if( refundToken1 > 0 ) {  
1581             require(IERC20(params.token1).transfer(_msgSender(), refundToken1));  
1582         }
```

2. Pre-attack preparation: the attacker 0x161ce...5899Fd first deployed the attack contract 0xCFFo7C4e6aa9E2fEco4DAaF5f41d1b1of3adAdF4 as well as created the token contract 0x2d4abfdcd1385951df4317f9f3463fb11b9a31df (token A).
3. The attacker calls the lockToken function in the LockToken contract, performs four locks of token A created by himself, and sets the withdrawal address to the attack contract address, and obtains four NFTs as LP (ids 15324, 15325, 15326, 15327).

same as the one being operated, and the parameter `sqrtPriceX96`, which is related to the price calculation of the UNI-V3 migration, is also input by the user.

```

  "_id" : "15324"
  "params" : {
    "amount0Min" : "0"
    "amount1Min" : "0"
    "deadline" : "1666859863"
    "fee" : "500"
    "liquidityToMigrate" : "15000000000000000000"
    "pair" : "0x854373387e41371ac6e307a1f29603c6fa10d872"
    "percentageToMigrate" : 1
    "recipient" : "0xba399a2580785a2ded740f5e30ec89fb3e617e6e"
    "refundAsETH" : true
    "tickLower" : "-100"
    "tickUpper" : "100"
    "token0" : "0x2d4abfdcd1385951df4317f9f3463fb11b9a31df"
    "token1" : "0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2"
  }
  "noLiquidity" : true
  "sqrtPriceX96" : "79210883607084793911461085816"
  "_mintNFT" : false
}
[OUTPUT] : "0x"
"gas" : {

```

FEG-WETH pair

token A

WETH

```

1531 function migrate(
1532     uint256 id,
1533     IV3Migrator.MigrateParams calldata params,
1534     bool noLiquidity,
1535     uint160 sqrtPriceX96,
1536     bool _mintNFT
1537 )
1538 external
1539 payable
1540 whenNotPaused
1541 nonReentrant
1542 {
1543     require(address(nonfungiblePositionManager) != address(0), "NFT manager not set");
1544     require(address(v3Migrator) != address(0), "v3 migrator not set");
1545     Items memory lockedERC20 = lockedToken[id];
1546     require(block.timestamp < lockedERC20.unlockTime, "Unlock time already reached");
1547     require(msgSender() == lockedERC20.withdrawalAddress, "Unauthorised sender");
1548     require(!lockedERC20.withdrawn, "Already withdrawn");
1549
1550     uint256 totalSupplyBeforeMigrate = nonfungiblePositionManager.totalSupply();
1551
1552     //scope for solving stack too deep error
1553     {
1554         uint256 ethBalanceBefore = address(this).balance;
1555         uint256 token0BalanceBefore = IERC20(params.token0).balanceOf(address(this));
1556         uint256 token1BalanceBefore = IERC20(params.token1).balanceOf(address(this));
1557
1558         //Initialize the pool if not yet initialized
1559         if(noLiquidity) {
1560             v3Migrator.createAndInitializePoolIfNecessary(params.token0, params.token1, params.fee, sqrtPriceX96);
1561         }
1562
1563         IERC20(params.pair).approve(address(v3Migrator), params.liquidityToMigrate);
1564
1565         v3Migrator.migrate(params);

```

Controllable parameters

6. The attack contract uses four NFTs prepared in advance to obtain the migration refunds for the four tokens: WETH, DAI, CAW, and TSUKA, all of which were sent to the 0xBa399a2580785A2dEd740F5e30EC89Fb3E617e6E address.

Address 0xBa399a2580785A2dEd740F5e30EC89Fb3E617e6E

Exploit

This address is reported to be involved in an [exploit on Team Finance](#). More details to follow.

Overview Team Finance Exploiter 2

Balance: 880.258661829593977641 Ether

Ether Value: \$1,324,155.50 (@ \$1,504.28/ETH)

Token: \$11,820,816.74 8

Search for Token Name

> ERC-20 Tokens (6)

Dai Stableco... (DAI)	\$6,429,025.48
6,429,327.65886799 DAI	@1.00
A Hunters Dr... (CAW)	\$4,660,294.25
74,613,657,577,043.9 CAW	@0.00
Dejitaru Tsu... (TSUKA)	\$731,491.28
11,837,577.7213003 TSUKA	@0.0618

Transactions Internal Txns

Latest 20 from a total of 20 transi

Vulnerability analysis

This attack mainly exploits a vulnerability in the migrate function of the LockToken contract. The validation of migrate is easily bypassed and can manipulate the price during migration.

Fund flow

The stolen funds are 880.258 ETH, 642,9327.6 DAI, 74,6136,5757,7043 CAW, 1183,7577.7 TSUKA, with a total value of about \$13 million, which remain in the attacker's address 0xBa399a2580785A2dEd740F5e30EC89Fb3E617e6E.

Summary

In response to this incident, Beosin security team recommends that

1. Carefully verify parameters of important functions, especially user-controllable parameters.
2. Choose a professional security audit company before the project goes live.

Contact

If you have need any blockchain security services, please contact us:

[Website](#) [Email](#) [Official Twitter](#) [Alert](#) [Telegram](#) [LinkedIn](#)