Security Audit Report for YPool Smart Contract

Date: Jan 26, 2022

Version: 1.2

Contact: contact@blocksecteam.com

BlockSec Audit Team

Digitally signed by BlockSec Audit Team
Date: 2022.01.27 00:04:01 +08'00'

Contents

Report Manifest

Item	Description
Client	XYPool
Target	YPool Smart Contract

Version History

Version	Date	Description
1.0	Jan 11, 2022	First Release
1.1	Jan 26, 2022	Status Update
1.2	Jan 26, 2022	Status Update

About BlockSec The BlockSec Team focuses on the security of the blockchain ecosystem, and collaborates with leading DeFi projects to secure their products. The team is founded by top-notch security researchers and experienced experts from both academia and industry. They have published multiple blockchain security papers in prestigious conferences, reported several zero-day attacks of DeFi applications, and released detailed analysis reports of high-impact security incidents. They can be reached at Email, Twitter and Medium.

Chapter 1 Introduction

1.1 About Target Contracts

Information	Description
Type	Smart Contract
Language	Solidity
Approach	Semi-automatic and manual verification

The auditing process is iterative. Specifically, we will audit the commits that fix the discovered issues. If there are new issues, we will continue this process. The commit SHA values of the repo [1] during the audit are shown in the following.

Contract Name	Stage	Commit SHA
YPool	Initial	74cc313fae9c1fa5a9a51c869c0add0876185e11
YPool	Final	73ca274046f4681c037d2d67b702ea70a0af15a6

1.2 Disclaimer

This audit report does not constitute investment advice or a personal recommendation. It does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Any entity should not rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset.

This audit report is not an endorsement of any particular project or team, and the report do not guarantee the security of any particular project. This audit does not give any warranties on discovering all security issues of the smart contracts, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit cannot be considered comprehensive, we always recommend proceeding with independent audits and a public bug bounty program to ensure the security of smart contracts.

The scope of this audit is limited to the code mentioned in Section 1.1. Unless explicitly specified, the security of the language itself (e.g., the solidity language), the underlying compiling toolchain and the computing infrastructure are out of the scope.

1.3 Procedure of Auditing

We perform the audit according to the following procedure.

- **Vulnerability Detection** We first scan smart contracts with automatic code analyzers, and then manually verify (reject or confirm) the issues reported by them.
- **Semantic Analysis** We study the business logic of smart contracts and conduct further investigation on the possible vulnerabilities using an automatic fuzzing tool (developed by our research team).

[1] https://github.com/XY-Finance/ypool-contracts-audit

We also manually analyze possible attack scenarios with independent auditors to cross-check the result.

- **Recommendation** We provide some useful advice to developers from the perspective of good programming practice, including gas optimization, code style, and etc.

We show the main concrete checkpoints in the following.

1.3.1 Software Security

- Reentrancy
- DoS
- Access control
- Data handling and data Flow
- Exception handling
- Untrusted external call and control flow
- Initialization consistency
- Events operation
- Error-prone randomness
- Improper use of the proxy system

1.3.2 DeFi Security

- Semantic consistency
- Functionality consistency
- Access control
- Business logic
- Token operation
- Emergency mechanism
- Oracle security
- Whitelist and blacklist
- Economic impact
- Batch transfer

1.3.3 NFT Security

- Duplicated item
- Verification of the token receiver
- Off-chain metadata security

1.3.4 Additional Recommendation

- Gas optimization
- Code quality and style

Note *The previous checkpoints are the main ones. We may use more checkpoints during the auditing process according to the functionality of the project.*

1.4 Security Model

To evaluate the risk, we follow the standards or suggestions that are widely adopted by both industry and academy, including OWASP Risk Rating Methodology [2] and Common Weakness Enumeration [3]. Accordingly, the severity measured in this report are classified into four categories: **High**, **Medium**, **Low** and **Undetermined**.

Furthermore, the status of a discovered issue will fall into one of the following four categories:

- **Undetermined** No response yet.
- **Acknowledged** The issue has been received by the client, but not confirmed yet.
- **Confirmed** The issue has been recognized by the client, but not fixed yet.
- **Fixed** The issue has been confirmed and fixed by the client.

[2] https://owasp.org/www-community/OWASP_Risk_Rating_Methodology

[3] https://cwe.mitre.org/

Chapter 2 Findings

In total, we find `three` potential issues in the smart contract. We also have `two` recommendations, as follows:

- High Risk: 0
- Medium Risk: 1
- Low Risk: 2
- Recommendations: 2

ID	Severity	Description	Category	Status
1	Low	Possible Loss with Incorrect Call Sequence	DeFi Security	Fixed
2	Medium	Arbitrary External Calls with NO Access Control	DeFi Security	Fixed
3	Low	Lack of Checks on Parameters of the `multiswap()` Function	DeFi Security	Confirmed
4	-	Fix Typos in Variables and Function Names	Recommendation	Fixed
5	-	Give Concrete Revert Messages	Recommendation	Fixed

The details are provided in the following sections.

2.1 DeFi Security

2.1.1 Possible Loss with Incorrect Call Sequence

Status Fixed

Description In `StakingRewardsAsync.sol`, a user may call the `exit()` function to withdraw the early stake. However, `stakerExists[msg.sender]` is set to `false` in the `completeExitRequest()` function (invoked by the stake worker). As such, if the user calls the `stake()` function prior to the invocation of `completeExitRequest()`, then he/she may lose the last stake.

```
 96    /// @notice Stake the specific amount of token (requred approval from address)
 97    /// @param amount The amount of token to stake
 98    function stake(uint256 amount) external {
 99        require(amount > 0, "ERR_INVALID_STAKE_AMOUNT");
100        stakeToken.safeTransferFrom(msg.sender, address(this), amount);
101        _totalSupply += amount;
102        _balances[msg.sender] += amount;
103        stakerExists[msg.sender] = true;
104        emit Staked(msg.sender, amount);
105    }
```

Listing 2.1: StakingRewardsAsync.sol

```
195    /// @notice Complete the exit request by returning staked token and transferring reward token
           (could be executed only by stake worker)
```

```
196     /// @param _nonce The request nonce to be processed
197     /// @param rewardAmount Amount of reward token to be minted to request account
198     function completeExitRequest(uint256 _nonce, uint256 rewardAmount) external onlyRole(
            ROLE_STAKE_WORKER) {
199         require(_nonce < nonce , "ERR_INVALID_NONCE");
200
201         Request storage request = requests[_nonce];
202         require(request.requestType == RequestType.Exit, "ERR_INVALID_REQUEST_TYPE");
203         require(request.status == RequestStatus.Pending, "ERR_REQUEST_NOT_PENDING");
204
205         request.status = RequestStatus.Completed;
206         stakeToken.safeTransfer(request.account, request.amount);
207         rewardToken.safeTransfer(request.account, rewardAmount);
208         stakerExists[request.account] = false;
209         emit ExitCompleted(_nonce, request.account, request.amount, rewardAmount);
210     }
```

Listing 2.2: StakingRewardsAsync.sol

Impact If a user accidentally calls these functions with incorrect call sequence, he/she may lose the last stake.

Suggestion N/A

2.1.2 Arbitrary External Calls with NO Access Control

Status Fixed

Description The `multiswap()` function of `Multicaller.sol` has no access control (i.e., everyone can call this function). What's more, there are no checks to verify the target contract and function being called (i.e., the target callee and the four bytes signature of the calldata for the `Call` struct).

```
67     function multiswap(Call[] memory calls, address payable receiver) public payable nonReentrant
           returns (uint256 blockNumber, bytes[] memory returnData) {
68         blockNumber = block.number;
69         returnData = new bytes[](calls.length);
70
71         // ‘calls‘ contain only swap actions
72         uint256 amountIn;
73         for(uint256 i = 0; i < calls.length; i++) {
74             approveIfNeeded(IERC20(calls[i].fromToken), calls[i].target);
75
76             if (calls[i].amountIn > 0) {
77                 amountIn = calls[i].amountIn;
78             }
79             uint256 value = 0;
80             bytes memory callData;
81             if (calls[i].fromToken == ETHER_ADDRESS) {
82                 value = amountIn;
83                 callData = abi.encodePacked(calls[i].prefix, calls[i].suffix);
84             } else {
85                 callData = abi.encodePacked(calls[i].prefix, amountIn, calls[i].suffix);
86             }
87             uint256 balance = uniBalanceOf(IERC20(calls[i].toToken), address(this));
```

```
88            bytes memory ret = Address.functionCallWithValue(calls[i].target, callData, value, "
                  Multicall multiswap: call failed");
89            amountIn = uniBalanceOf(IERC20(calls[i].toToken), address(this)) - balance;
90            returnData[i] = ret;
91        }
92...
```

Listing 2.3: MultiCaller.sol

Impact The insufficient access control and parameter checks may lead to potential problems. For example, an external bot may be able to harvest funds that are stored in this contract.

Suggestion Add corresponding access control and sanity checks.

2.1.3 Lack of Checks on Parameters of the `multiswap()` Function

Status Confirmed

Description The function `multiswap()` in `Multicaller.sol` performs insufficient check with the parameters. Specifically,

1. No checks if the swaps in `calls` are continuous, i.e., `calls[i].fromToken == calls[i-1].toToken`.
2. The `amountIn` specified in the call parameter takes priority for constructing calls. If `amountIn` in the `calls` passed as parameters is incorrect, it may result in a loss of funds froze in this contract or a failure of the swap.

```
67    function multiswap(Call[] memory calls, address payable receiver) public payable nonReentrant
          returns (uint256 blockNumber, bytes[] memory returnData) {
68        blockNumber = block.number;
69        returnData = new bytes[](calls.length);
70
71        // ‘calls‘ contain only swap actions
72        uint256 amountIn;
73        for(uint256 i = 0; i < calls.length; i++) {
74            approveIfNeeded(IERC20(calls[i].fromToken), calls[i].target);
75
76            if (calls[i].amountIn > 0) {
77                amountIn = calls[i].amountIn;
78            }
79            uint256 value = 0;
80            bytes memory callData;
81            if (calls[i].fromToken == ETHER_ADDRESS) {
82                value = amountIn;
83                callData = abi.encodePacked(calls[i].prefix, calls[i].suffix);
84            } else {
85                callData = abi.encodePacked(calls[i].prefix, amountIn, calls[i].suffix);
86            }
87            uint256 balance = uniBalanceOf(IERC20(calls[i].toToken), address(this));
88            bytes memory ret = Address.functionCallWithValue(calls[i].target, callData, value, "
                  Multicall multiswap: call failed");
89            amountIn = uniBalanceOf(IERC20(calls[i].toToken), address(this)) - balance;
90            returnData[i] = ret;
91        }
92...
```

Listing 2.4: MultiCaller.sol

Impact Combining with issue 2.1.2, funds frozen in this contract can be harvested by external attackers or bots.

Suggestion Add more sanity checks on parameters `calls`.

Feedback from the Developers On one hand, we don't check `amountIn` in the calls. Because for continuous calls `calls[i]` and `calls[i+1]`, we can only determine `calls[i+1].amountIn` after `calls[i]` is actually executed. Instead, we verify that the swap is successful in `Aggregator.swap()` by checking the total amount out from `Multicaller.multiswap()` is greater than or equal to `minReturnAmount`. Apart from that, we also require the caller of `Multicaller.multiswap()` must be an Aggregator. On the other hand, we cannot add the check to ensure the continuous sequence of the swap, i.e., `require(calls[i].toToken == calls[i+1].fromToken)`. This is because in some scenarios we might have split routes aggregated for a swap.

2.2 Additional Recommendation

2.2.1 Fix Typos in Variables and Function Names

Status Fixed

Description There are some mis-spelling in variable and function names. For example, `YPoolSupoortedToken` and `setYPoolSupoortedToken()` in `RebalanceRewardsAsync.sol` (notice the extraneous "o" in the names).

```
43    // mappings of YPool tokens that are allowed to request redeem reward
44    mapping (address => bool) public YPoolSupoortedToken;
```

Listing 2.5: RebalanceRewardsAsync.sol

```
74    /// @notice Set the pool token allowed to sent redeem request (could be set only by manager)
75    /// @param _supportedToken The YPool supported token
76    /// @param isSet Set to enable or disable
77    function setYPoolSupoortedToken(address _supportedToken, bool isSet) external onlyRole(
          ROLE_MANAGER) {
78        YPoolSupoortedToken[_supportedToken] = isSet;
79    }
```

Listing 2.6: RebalanceRewardsAsync.sol

Impact N/A

Suggestion Fix these typos.

2.2.2 Give Concrete Revert Messages

Status Fixed

Description Some functions do not have clear revert messages, e.g., `XSwapper.getSwapRequest()`.

```
215    /// @notice Get a certain swap request
216    /// @param _swapId Swap Id of a swap request
217    function getSwapRequest(uint256 _swapId) external view returns (SwapRequest memory) {
218        if (_swapId >= swapId) revert();
219        return swapRequests[_swapId];
220    }
```

Listing 2.7: XSwapper.sol

Impact N/A

Suggestion Add corresponding revert messages.