Security Audit of Poolz Finance's Smart Contracts

a report of findings by

Arcadia

innovative fortuna iuvat

May 17th, 2021

Table of Contents

Document Info **1**

Contact **2**

Executive Summary **3**

Findings **4**

Safe transfer token for function WithdrawToken 4

Functions doing the same thing 6

Conclusion **7**

Disclaimer **7**

innovative fortuna iuvat 0

Document Info

Client	Poolz Finance
Title	Security Audit of Poolz Finance’s Smart Contracts
Approved By	Rasikh Morani

Contact

For more information on this report, contact The Arcadia Media Group Inc.

Rasikh Morani
(972) 543-3886
rasikh@arcadiamgroup.com
https://t.me/thearcadiagroup

Executive Summary

A Representative Party of Poolz Finance ("Poolz") engaged The Arcadia Group ("Arcadia"), a software development, research, and security company, to conduct a review of the following Poolz Finance smart contracts on 4 github repositories:

- Locked Pools at Commit #25a58d67119fc8b3c9d32d885639e7c6c8294290.
- Poolz Helper at commit #cda1aa52132a68b7c6f0aa3b0310c270422383af
- Benefit at commit #a944b246a09fbcdeff44c2c6d7b24944ac9ea79d
- Whitelist at commit #a4b8dfabe778ff603f145a7f374a20a242f1fc24

Moreover, Arcadia also reviewed two pull requests at PR6 and PR9 which are related to code refactoring.

Arcadia completed this security review using various methods primarily consisting of dynamic and static analysis. This process included a line-by-line analysis of the in-scope contracts, optimization analysis, analysis of key functionalities and limiters, and reference against intended functionality.

There were 02 issues found, 00 of which were deemed to be ‘critical’, and 00 of which were rated as ‘high’.

Severity Rating	Number Of Original Occurrences	Number Of Remaining Occurrences
Critical	00	00
High	00	00
Medium	01	01
Low	01	01
Notice	00	00
Informational	00	00

Findings

1. Safe transfer token for function WithdrawToken

- PLZ-1
- Severity: Medium
- Likelihood: Medium
- Impact: Low

- Target: LockedDeal.sol
- Category: WithdrawToken
- Finding Type: Dynamic

In `LockedDeal` contract, function `WithdrawToken` should check whether the contract has sufficient token amount to withdraw. The function can be reverted if there is less than `AllPoolz[_PoolId].Amount` token in the contract.

```
function WithdrawToken(uint256 _PoolId) public returns (bool) {
      //pool is finished + got left overs + did not took them
      if (
          _PoolId < Index &&
          AllPoolz[_PoolId].UnlockTime <= now &&
          AllPoolz[_PoolId].Amount > 0
      ) {
          TransferToken(
              AllPoolz[_PoolId].Token,
              AllPoolz[_PoolId].Owner,
              AllPoolz[_PoolId].Amount
          );
          AllPoolz[_PoolId].Amount = 0;
          return true;
      }
      return false;
  }
```

Action Recommended: Verify whether the contract has enough token amount to withdraw. The following code is a possible suggestion:

```
function WithdrawToken(uint256 _PoolId) public returns (bool) {
       //pool is finished + got left overs + did not took them
       if (
           _PoolId < Index &&
           AllPoolz[_PoolId].UnlockTime <= now &&
           AllPoolz[_PoolId].Amount > 0
       ) {
           uint256 contractBalance =
IERC20(AllPoolz[_PoolId].token).balanceOf(address(this));
           uint256 amount = contractBalance > AllPoolz[_PoolId].Amount?
AllPoolz[_PoolId].Amount: contractBalance;
           TransferToken(
               AllPoolz[_PoolId].Token,
               AllPoolz[_PoolId].Owner,
               AllPoolz[_PoolId].Amount
           );
           AllPoolz[_PoolId].Amount = 0;
           return true;
       }
       return false;
   }
```

2. Functions appear to be doing the same thing

- PLZ-2
- Severity: Low
- Impact: Low
- Target: LockedPoolz.sol
- Category: Functionality duplication
- Finding Type: Dynamic

Functions `CreateMassPools` and `CreatePoolsWrtTime` do almost the same thing.

```
function CreateMassPools(
        address _Token,
        uint64[] calldata _FinishTime,
        uint256[] calldata _StartAmount,
        address[] calldata _Owner
    ) external isGreaterThanZero(_Owner.length) isBelowLimit(_Owner.length)
returns(uint256, uint256) {
        // require(_Owner.length <= maxTransactionLimit, "Array length Invalid");
        require(_Owner.length == _FinishTime.length, "Date Array Invalid");
        require(_Owner.length == _StartAmount.length, "Amount Array Invalid");
        uint256 firstPoolId = Index;
        for(uint i=0 ; i < _Owner.length; i++){
            CreateNewPool(_Token, _FinishTime[i], _StartAmount[i], _Owner[i]);
        }
        uint256 lastPoolId = SafeMath.sub(Index, 1);
        return (firstPoolId, lastPoolId);
    }

    // create pools with respect to finish time
    function CreatePoolsWrtTime(
        address _Token,
        uint64[] calldata _FinishTime,
        uint256[] calldata _StartAmount,
        address[] calldata _Owner
    )    external
        isGreaterThanZero(_Owner.length)
        isGreaterThanZero(_FinishTime.length)
        isBelowLimit(_Owner.length * _FinishTime.length)
        returns(uint256, uint256)
    {
```

```
        require(_Owner.length * _FinishTime.length <= maxTransactionLimit, "Array
length Invalid");
        require(_Owner.length == _StartAmount.length, "Amount Array Invalid");
        uint256 firstPoolId = Index;
        for(uint i=0 ; i < _FinishTime.length ; i++){
            for(uint j=0 ; j < _Owner.length ; j++){
                CreateNewPool(_Token, _FinishTime[i], _StartAmount[j], _Owner[j]);
            }
        }
        uint256 lastPoolId = SafeMath.sub(Index, 1);
        return (firstPoolId, lastPoolId);
    }
```

Action Recommended: These seem to be redundant code. It is advisable to re-define the functionality of one of the two functions.

Conclusion

Arcadia identified issues that occurred at 4 different repositories:

- Locked Pools at Commit #25a58d67119fc8b3c9d32d885639e7c6c8294290.
- Poolz Helper at commit #cda1aa52132a68b7c6f0aa3b0310c270422383af
- Benefit at commit #a944b246a09fbcdeff44c2c6d7b24944ac9ea79d
- Whitelist at commit #a4b8dfabe778ff603f145a7f374a20a242f1fc24

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, The Arcadia Group and the Authors assume no responsibility for errors, omissions, or damages resulting from the use of the provided information. Additionally, Arcadia would like to emphasize that the use of Arcadia's services does not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One audit on its own is not enough for a project to be considered secure; that categorization can only be earned through extensive peer review and battle testing over an extended period.