THE ARCADIA GROUP

Security Audit Report

Poolz V2

9/3/2022

Revision: 1/5/2023

PREPARED FOR:
Poolz, Poolz.Finance

ARCADIA CONTACT INFO
Email: audits@arcadiamgroup.com
Telegram: https://t.me/thearcadiagroup

Table of Contents

Executive Summary

A Representative Party of POOLZ ("**CLIENT**") engaged The Arcadia Group ("Arcadia"), a software development, research, and security company, to conduct a review of the following Poolz smart contracts on the **The-Poolz/Locked-pools** github repository at Commit **#71c2d4b742a90bdb1556f5fc8836bd16785c01b8**

The scope of this audit included the following files:

1. **LockedControl**.sol
2. **LockedCreation**.sol
3. **LockedDealEvents.**sol
4. **LockedDealModifiers**.sol
5. **LockedDealV2**.sol
6. **LockedManageable**.sol
7. **LockedPoolz**.sol
8. **LockedPoolzData**.sol

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 **6** issues found, **0** of which were deemed to be 'critical', and **1** of which were rated as 'high'.

Severity Rating	Number of Original Occurrences	Number of Remaining Occurrences
CRITICAL	0	0
HIGH	2	0
MEDIUM	3	1
LOW	1	1
INFORMATIONAL	0	0

Findings

1. Use of tx.origin

Issue: **POOLZ V2-1**
Severity: **High**

Target:**FeeBaseHelper.sol, LockedCreation.sol**
Finding Type: **DYNAMIC**

The `LockedCreation` contract depends on the `FeeBaseHelper` contract of the library Poolz-Helper. FeeBaseHelper uses tx.origin which could be made vulnerable to the LockedCreation contract as in SWC-115.

Action Recommended:

As in SWC-115, using `msg.sender` in FeeBaseHelper for best practice.

2. PoolTransfer function requires transfer-in

Issue: **POOLZ V2-2**
Severity: **High**

Target:**LockedControl.sol**
Finding Type: **DYNAMIC**

The function is to transfer the ownership of a pool owner to a new owner.

The current code does it by

- creating a new pool for the new pool owner, and
- setting the `DebitedAmount` field of the new pool to the `DebitedAmount` field of the old pool

```
function PoolTransfer(uint256 _PoolId, address _NewOwner)
       external
       isPoolValid(_PoolId)
       isPoolOwner(_PoolId)
       notZeroAddress(_NewOwner)
   {
       Pool storage pool = AllPoolz[_PoolId];
```

```
        require(_NewOwner != pool.Owner, "Can't be the same owner");
        require(
            pool.FinishTime > block.timestamp,
            "Can't create with past finish time"
        );
        uint256 newPoolId = CreatePool(
            pool.Token,
            pool.StartTime,
            pool.FinishTime,
            pool.StartAmount,
            _NewOwner
        );
        pool.StartAmount = 0;
        AllPoolz[newPoolId].DebitedAmount = pool.DebitedAmount;
        pool.DebitedAmount = 0;
        emit PoolTransferred(newPoolId, _PoolId, _NewOwner, msg.sender);
    }
```

The functionality of this function is confused as it is equivalent to creating a new pool and nullify the old pool.

Action Recommended:

- Simple change the `Owner` field to the `_NewOwner` variable without creating a new pool

3. Missing check for _NewOwner not zero

Issue: **POOLZ V2-3**
Severity: **MEDIUM**

Target:**LockedControl.sol**
Finding Type: **DYNAMIC**

The `SplitPoolAmount` function calls the `SplitPool` function to create a new pool with an input variable `_NewOwner`. However, `_NewOwner` is not checked for a non-zero address.

Action Recommended:

Add `notZeroAddress` modifier check for `SplitPoolAmount` function.

4. Public view functions should not use `msg.sender`

Issue: **POOLZ V2-4**
Severity: Medium

Target:**LockedPoolzData.sol**
Finding Type: **DYNAMIC**

The public view functions `GetAllMyPoolsId`, `GetMyPoolsId`, `GetMyPoolsIdByToken`, `GetMyPoolDataByToken` should not use `msg.sender` as most view functions are not called directly by a user wallet, but through an RPC provider, i.e. Infura.

For example, when a user reads data from smart contracts through calling these functions on EtherScan, returned results will be mostly empty as the msg.sender in these functions won't be the user wallet address.

Action Recommended:

It is necessary to make use of an input parameter `_UserAddress` to pass to the view functions as the below code example.

```
function GetAllMyPoolsId(address _UserAddress) public view returns (uint256[]
memory) {
       return MyPoolz[_UserAddress];
   }
```

Findings for V2.3.0

The following describes the list of issues for the version V2.3.0 of the contracts.

5. Function `ApproveAllowance` should not have the `notZeroValue` modifier

Issue: **POOLZ V2-5**
Severity: **Medium**

Target:**LockedControl.sol**
Finding Type: **DYNAMIC**

The function is for a pool owner to change the `allowance` amount for delegating a `spender` to be able to split the pool into pools.

Semantically, the function should also allow the pool owner to set the allowance amount for a spender to zero, which disables the allowance of the spender.

The current implementation, however, imposes the `notZeroValue`, which prohibits the ability of setting allowance amount to zero.

The current code does it by

- creating a new pool for the new pool owner, and
- setting the `DebitedAmount` field of the new pool to the `DebitedAmount` field of the old pool

```
function ApproveAllowance(
        uint256 _PoolId,
        uint256 _Amount,
        address _Spender
    )
        external
        isPoolValid(_PoolId)
        isPoolOwner(_PoolId)
        notZeroValue(_Amount)
        notZeroAddress(_Spender)
    {
        Allowance[_PoolId][_Spender] = _Amount;
        emit PoolApproval(_PoolId, _Spender, _Amount);
    }
```

THE ARCADIA GROUP

Action Recommended:

- The implementation of `ApproveAllowance` function should remove the `notZeroValue` modifier

6. Function `CreatePool` should check `_CliffTime`

Issue: **POOLZ V2-6**
Severity: **Low**

Target:**LockedPoolz.sol**
Finding Type: **DYNAMIC**

The function should ensure that the `_CliffTime >= _FinishTime` and `_CliffTime > block.timestamp` so that the pool does not end up at letting any one to call the function to release token to the pool owner address before the pool finish.

Action Recommended:

- Add additional require statements for `_CliffTime >= _FinishTime` and `_CliffTime > block.timestamp` in function `CreatePool`

Conclusion

Arcadia identified issues that occurred at hash **#71c2d4b742a90bdb1556f5fc8836bd16785c01b8**.

Further issues POOLZ V2-5 and POOLZ V2-6 were identified for the version V2.3.0 of the contracts.

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.