Security Audit Report

Poolz LockDealNFT Dispenser Provider

12/10/2024

Revision: 12/19/2024

PREPARED FOR:
Poolz

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

Table of Contents

Executive Summary **2**

1. Introduction and Audit Scope 2

2. Audit Summary 2

Findings in Manual Audit **4**

1. Confusing modifier name 4

2. Modifier isCallerApproved can be bypassed 6

3. Modifier isValidSignature may create bad user experience 8

4. NFTs created by the contract without events emission 10

5. Hardcoded unknown function selectors 12

Automated Audit **16**

Static Analysis with Slither 16

Unit tests 16

Conclusion **18**

Disclaimer **18**

Executive Summary

1. Introduction and Audit Scope

A Representative of Poolz ("**CLIENT**") engaged Arcadia, a software development, research, and security company, to conduct a review of the following Poolz smart contracts on the following described audit scope:

- **LockDealNFT Dispenser Provider contracts at commits**
    - **#92eb746e1f46b37a986a8f86b2b06ddda6adf002**

2. Audit Summary

a. Audit Methodology

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.

The followings are the steps we have performed while auditing the smart contracts:

- Investigating the project and its technical architecture overview through its documentation
- Understanding the overview of the smart contracts, the functions of the contracts, the inheritance, and how the contracts interface with each others thanks to the graph created by Solidity Visual Developer
- Manual smart contract audit:
    - Review the code to find any issue that could be exploited by known attacks listed by Consensys
    - Identifying which existing projects the smart contracts are built upon and what are the known vulnerabilities and remediations to the existing projects
    - Line-by-line manual review of the code to find any algorithmic and arithmetic related vulnerabilities compared to what should be done based on the project's documentation
    - Find any potential code that could be refactored to save gas
    - Run through the unit-tests and test-coverage if exists

- Automated smart contract audit:
  - Scanning for vulnerabilities in the smart contracts using Static Code Analysis Software
  - Making a static analysis of the smart contracts using Slither
- Additional review: a follow-up review is done when the smart contracts have any new update. The follow-up is done by reviewing all changes compared to the audited commit revision and its impact to the existing source code and found issues.

b. Summary

There were **5** 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	1	0
MEDIUM	1	0
LOW	2	0
INFORMATIONAL	1	0

Findings in Manual Audit

1. Confusing modifier name

Issue ID

LDP-1

Risk level

Severity: Low, likelihood: Low

Code segment

```
/// @notice Ensures that the tokens have not already been taken for the specified
pool and owner.
   /// @dev Reverts with a `TokensAlreadyTaken` error if tokens have already been
dispensed for the given pool and owner.
   /// @param poolId The pool ID to check.
   /// @param owner The owner to verify if tokens have already been dispensed.
   modifier isAlreadyTaken(uint256 poolId, address owner) {
       if (isTaken[poolId][owner]) {
           revert TokensAlreadyTaken(poolId, owner);
       }
       _;
   }
```

Description

The modifier checks whether the token in the pool has not been dispensed by checking the state mapping variable isTaken. The modifier’s name, however means to ensure that the token in the pool has already been dispensed, which is opposite to the modifier's functioning.

Code location

```
contracts/DispenserModifiers.sol
```

Recommendation

Refactoring by renaming the modifier to `isNotAlreadyTaken`.

Remediation status

Resolved at #170f9d6e66baaec9e5fadc54a572af9712bcf6da

2. Modifier isCallerApproved can be bypassed

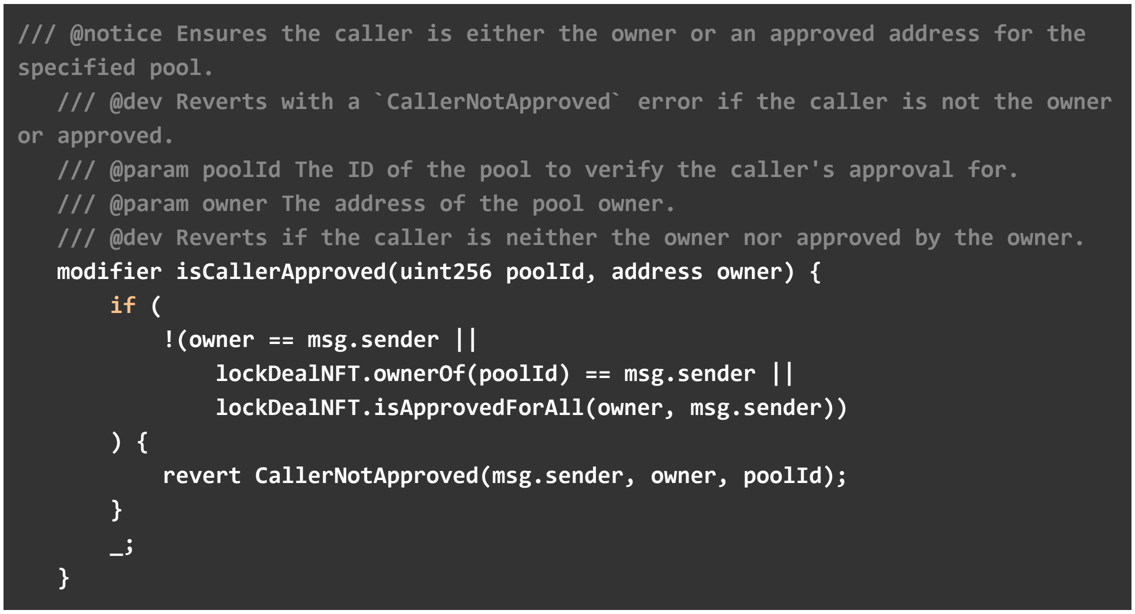
Issue ID

LDP-2

Risk level

Severity: High

Code segment



```
/// @notice Ensures the caller is either the owner or an approved address for the
specified pool.
   /// @dev Reverts with a `CallerNotApproved` error if the caller is not the owner
or approved.
   /// @param poolId The ID of the pool to verify the caller's approval for.
   /// @param owner The address of the pool owner.
   /// @dev Reverts if the caller is neither the owner nor approved by the owner.
   modifier isCallerApproved(uint256 poolId, address owner) {
       if (
           !(owner == msg.sender ||
               lockDealNFT.ownerOf(poolId) == msg.sender ||
               lockDealNFT.isApprovedForAll(owner, msg.sender))
       ) {
           revert CallerNotApproved(msg.sender, owner, poolId);
       }
       _;
   }
```

Description

The modifier ensures that the caller is either the owner or an approved operator for the specified pool. The modifier should revert if the caller is not the pool owner or an approved operator.
The modifier's code can be however bypassed by having the owner as the caller (`msg.sender`). Having owner = msg.sender can bypass the check in the `if` statement, thus bypassing the modifier, which then can cause unexpected damage to the pools.

Code location

```
src/DispenserModifiers.sol
```

Recommendation

The modifier should only bypass if either the following is true:

- The owner of the pool is msg.sender and msg.sender == owner
- Or the owner of the pool is owner and msg.sender is approved

The modifier can be rewritten as follows:

```
/// @notice Ensures the caller is either the owner or an approved
address for the specified pool.
   /// @dev Reverts with a `CallerNotApproved` error if the caller
is not the owner or approved.
   /// @param poolId The ID of the pool to verify the caller's
approval for.
   /// @param owner The address of the pool owner.
   /// @dev Reverts if the caller is neither the owner nor approved
by the owner.
   modifier isCallerApproved(uint256 poolId, address owner) {
       if (
           !((owner == msg.sender && lockDealNFT.ownerOf(poolId) ==
msg.sender) ||
               (lockDealNFT.ownerOf(poolId) == owner &&
               lockDealNFT.isApprovedForAll(owner, msg.sender)))
       ) {
           revert CallerNotApproved(msg.sender, owner, poolId);
       }
       _;
   }
```

Remediation status

Resolved at #170f9d6e66baaec9e5fadc54a572af9712bcf6da

3. Modifier isValidSignature may create bad user experience

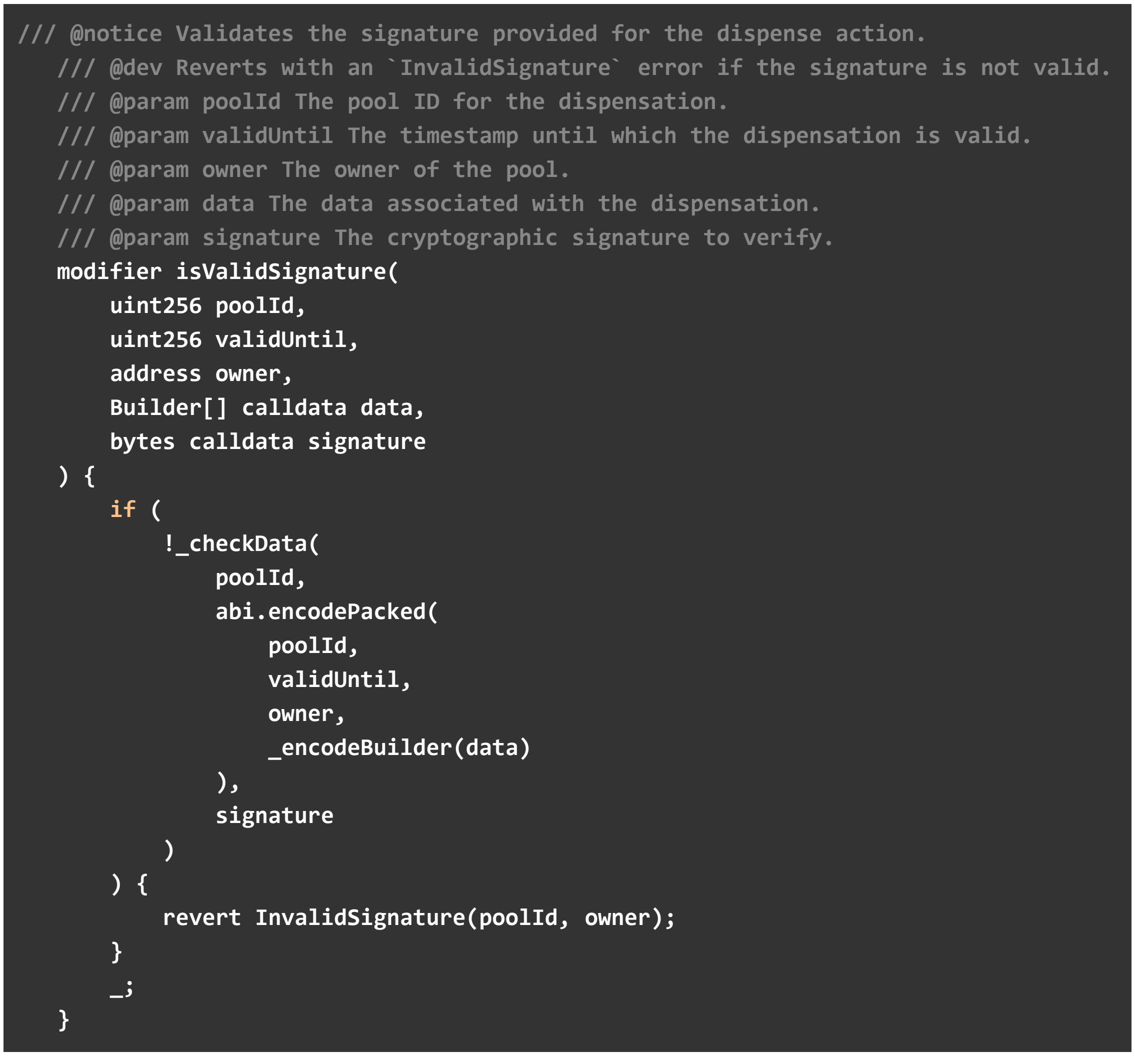
Issue ID

LDP-3

Risk level

Severity: Medium

Code segment



```
/// @notice Validates the signature provided for the dispense action.
    /// @dev Reverts with an `InvalidSignature` error if the signature is not valid.
    /// @param poolId The pool ID for the dispensation.
    /// @param validUntil The timestamp until which the dispensation is valid.
    /// @param owner The owner of the pool.
    /// @param data The data associated with the dispensation.
    /// @param signature The cryptographic signature to verify.
    modifier isValidSignature(
        uint256 poolId,
        uint256 validUntil,
        address owner,
        Builder[] calldata data,
        bytes calldata signature
    ) {
        if (
            !_checkData(
                poolId,
                abi.encodePacked(
                    poolId,
                    validUntil,
                    owner,
                    _encodeBuilder(data)
                ),
                signature
            )
        ) {
            revert InvalidSignature(poolId, owner);
        }
        _;
    }
```

Description

The modifier is for ensuring that the signature is created by the owner of the pool. In case of the transaction created by the owner of the pool, the transaction creator need to confirm their wallet two times:

- To sign for the data to create signature
- To confirm to send the transaction

These two times for confirmation may cause bad user experience to users.

Code location

```
src/DispenserModifiers.sol
```

Recommendation

To not create uncomfortable user experiences when pool owners create transactions for dispensing pool tokens, it is recommended to consider bypassing the modifier if the pool owner is the transaction sender.

The modifier can be rewritten as follows:

```
modifier isValidSignature(
       uint256 poolId,
       uint256 validUntil,
       address owner,
       Builder[] calldata data,
       bytes calldata signature
   ) {
       if (
           !(lockDealNFT.getData(poolId).owner == msg.sender ||
_checkData(
               poolId,
               abi.encodePacked(
                   poolId,
                   validUntil,
                   owner,
                   _encodeBuilder(data)
               ),
               signature
           ))
       ) {
           revert InvalidSignature(poolId, owner);
       }
       _;
   }
```

Remediation status

Resolved at #170f9d6e66baaec9e5fadc54a572af9712bcf6da

4. NFTs created by the contract without events emission

Issue ID

LDP-4

Risk level

Severity: Informational, Likelihood: Low

Code segment

```
/// @notice Iterates through the NFTs and dispenses them from the
pool.
    /// @dev Ensures that the amount taken is greater than 0 and
performs the necessary actions to create and withdraw NFTs.
    /// @param tokenPoolId The unique identifier for the token pool.
    /// @param owner The address of the owner requesting to dispense
tokens.
    /// @param data The Builder struct containing the data for the
NFT to be dispensed.
    /// @return amountTaken The amount of tokens dispensed for this
NFT.
    function _nftIterator(
        uint256 tokenPoolId,
        address owner,
        Builder calldata data
    ) internal firewallProtectedSig(0x592181eb) returns (uint256
amountTaken) {
        amountTaken = data.params[0]; // calling function must check
for an array of non-zero length
        if (amountTaken == 0) {
            revert AmountMustBeGreaterThanZero();
        }
        uint256 poolId = _createSimpleNFT(tokenPoolId, owner, data);
        if (lockDealNFT.isApprovedForAll(owner, address(this))) {
            _withdrawIfAvailable(poolId);
        }
    }

    /// @notice Creates a simple NFT for a given pool and owner.
    /// @param tokenPoolId The unique identifier for the token pool.
    /// @param owner The address of the owner requesting to mint the
NFT.
    /// @param data The Builder struct containing the data for the
NFT to be minted.
    /// @return poolId The unique identifier of the minted NFT.
    function _createSimpleNFT(
        uint256 tokenPoolId,
        address owner,
        Builder calldata data
    ) internal firewallProtectedSig(0xe64fbb17) returns (uint256
poolId) {
        poolId = lockDealNFT.mintForProvider(owner,
data.simpleProvider);
        data.simpleProvider.registerPool(poolId, data.params);
        lockDealNFT.cloneVaultId(poolId, tokenPoolId);
    }
```

Description

The functions create an NFT for a given pool and owner. These functions lack event emissions when NFTs are created by the dispenser contract. These events play a critical role for notifying relevant entities about changes in state variables and help inform off-chain systems about these changes without requiring complex querying or data extraction from the blockchain.

Code location

```
contracts/DispenserProvider.sol
```

Recommendation

It is recommended to have event emissions in the above NFT-create functions

Remediation status

Resolved at #170f9d6e66baaec9e5fadc54a572af9712bcf6da

5. Hardcoded unknown function selectors

Issue ID

LDP-5

Risk level

Severity: Low, Likelihood: Low

Code segment

```
/
// @notice Iterates through the NFTs and dispenses them from the pool.
   /// @dev Ensures that the amount taken is greater than 0 and performs the necessary actions to create and withdraw NFTs.
   /// @param tokenPoolId The unique identifier for the token pool.
   /// @param owner The address of the owner requesting to dispense tokens.
   /// @param data The Builder struct containing the data for the
NFT to be dispensed.
   /// @return amountTaken The amount of tokens dispensed for this
NFT.
   function _nftIterator(
       uint256 tokenPoolId,
       address owner,
       Builder calldata data
   ) internal firewallProtectedSig(0x592181eb) returns (uint256
amountTaken) {
       amountTaken = data.params[0]; // calling function must check
for an array of non-zero length
       if (amountTaken == 0) {
           revert AmountMustBeGreaterThanZero();
       }
       uint256 poolId = _createSimpleNFT(tokenPoolId, owner, data);
       if (lockDealNFT.isApprovedForAll(owner, address(this))) {
           _withdrawIfAvailable(poolId);
       }
   }

   /// @notice Finalizes the deal by ensuring the dispensed amount
does not exceed the available tokens in the pool.
   /// @dev Updates the pool amount and marks the transaction as
completed for the owner.
   /// @param tokenPoolId The unique identifier for the token pool.
   /// @param owner The address of the owner requesting to dispense
tokens.
   /// @param amountTaken The total amount of tokens dispensed from
the pool.
   function _finalizeDeal(
       uint256 tokenPoolId,
       address owner,
       uint256 amountTaken
   ) internal firewallProtectedSig(0x52f83cd6) {
       if (amountTaken > poolIdToAmount[tokenPoolId]) {
           revert NotEnoughTokensInPool(amountTaken,
poolIdToAmount[tokenPoolId]);
       }
```

```
        poolIdToAmount[tokenPoolId] -= amountTaken;
        isTaken[tokenPoolId][owner] = true;
    }
```

Description

The `firewallProtectedSig` modifier has hardcoded input parameters of function selectors 0x592181eb and 0x52f83cd6 without comments explaining the usage of the selectors.

Hardcoded code without comments has several significant downsides that can affect maintainability, readability, and functionality. Here's a breakdown of the key issues:

- The function selectors in the modifier signify which functions will be called in the modifier code execution, which may cause unexpected issues.

```
modifier firewallProtectedSig(bytes4 selector) {
       address firewall = _getAddressBySlot(FIREWALL_STORAGE_SLOT);
       if (firewall == address(0)) {
           _;
           return;
       }
       uint256 value = _msgValue();
       IFirewall(firewall).preExecution(_msgSender(), abi.encodePacked(selector),
value);
       _;
       IFirewall(firewall).postExecution(_msgSender(), abi.encodePacked(selector),
value);
   }
```

- Hardcoded values (e.g., numbers, strings, paths) provide no explanation for their purpose or origin. Without comments, it's difficult to understand why a specific value was chosen or how it fits into the overall logic.
- Future developers (or even the original author) may struggle to interpret the intent behind the code, leading to wasted time.
- Hidden dependencies: Hardcoded values can cause issues when they depend on external factors or are used inconsistently.

Code location

```
contracts/DispenserInternal.sol
```

Recommendation

It is recommended to have clear comments on what is the meaning of the hardcoded values and why are these values used.

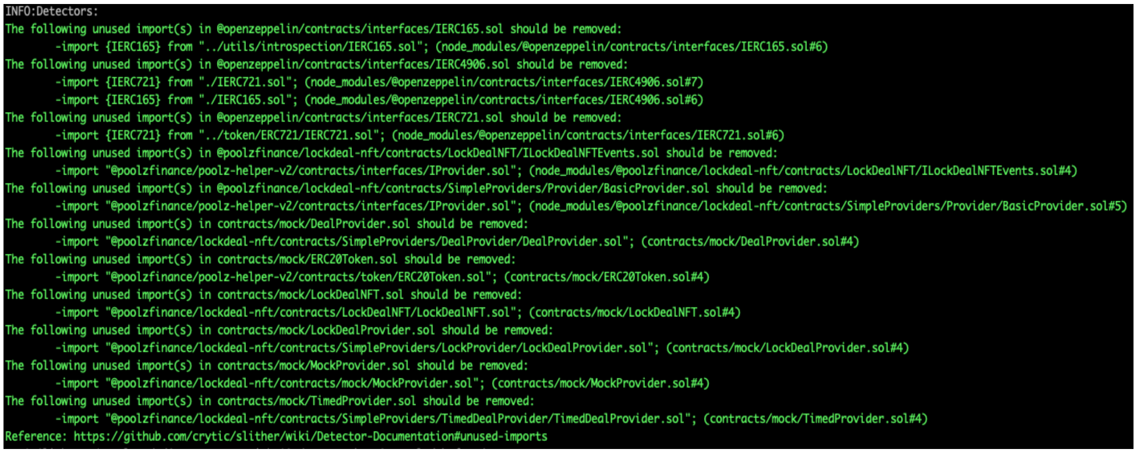
Remediation status

Resolved at #170f9d6e66baaec9e5fadc54a572af9712bcf6da

Automated Audit

Static Analysis with Slither

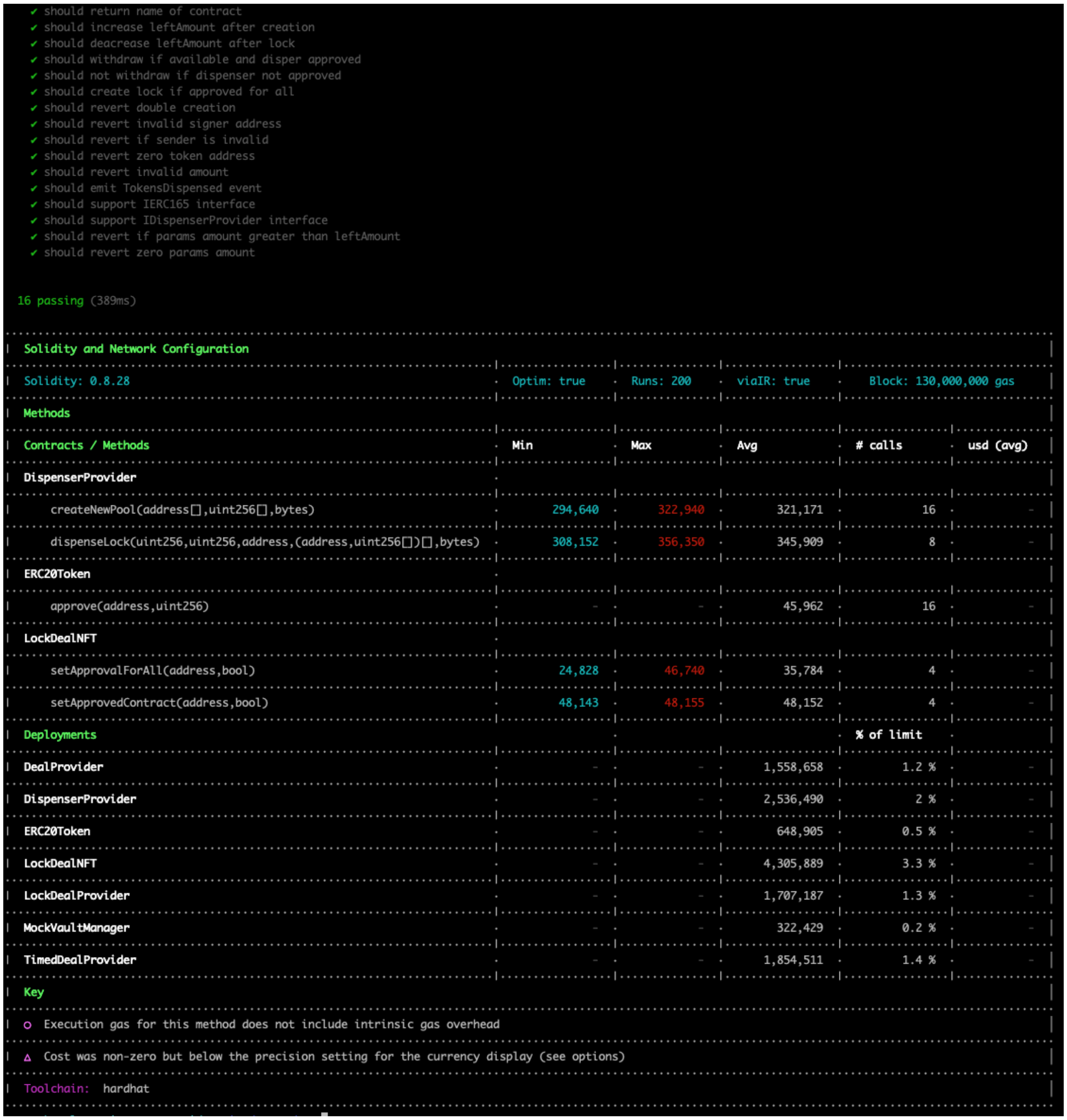
We run a static analysis against the source code using Slither, which is a Solidity static analysis framework written in Python 3. Slither runs a suite of vulnerability detectors, prints visual information about contract details. Slither enables developers to find vulnerabilities, enhance their code comprehension, and quickly prototype custom analyses. Our static analysis does not detect any further vulnerabilities from the contracts.



```
INFO:Detectors:
The following unused import(s) in @openzeppelin/contracts/interfaces/IERC165.sol should be removed:
	-import {IERC165} from "../utils/introspection/IERC165.sol"; (node_modules/@openzeppelin/contracts/interfaces/IERC165.sol#6)
The following unused import(s) in @openzeppelin/contracts/interfaces/IERC4906.sol should be removed:
	-import {IERC721} from "./IERC721.sol"; (node_modules/@openzeppelin/contracts/interfaces/IERC4906.sol#7)
	-import {IERC165} from "./IERC165.sol"; (node_modules/@openzeppelin/contracts/interfaces/IERC4906.sol#6)
The following unused import(s) in @openzeppelin/contracts/interfaces/IERC721.sol should be removed:
	-import {IERC721} from "../token/ERC721/IERC721.sol"; (node_modules/@openzeppelin/contracts/interfaces/IERC721.sol#6)
The following unused import(s) in @poolzfinance/lockdeal-nft/contracts/LockDealNFT/ILockDealNFTEvents.sol should be removed:
	-import "@poolzfinance/poolz-helper-v2/contracts/interfaces/IProvider.sol"; (node_modules/@poolzfinance/lockdeal-nft/contracts/LockDealNFT/ILockDealNFTEvents.sol#4)
The following unused import(s) in @poolzfinance/lockdeal-nft/contracts/SimpleProviders/Provider/BasicProvider.sol should be removed:
	-import "@poolzfinance/poolz-helper-v2/contracts/interfaces/IProvider.sol"; (node_modules/@poolzfinance/lockdeal-nft/contracts/SimpleProviders/Provider/BasicProvider.sol#5)
The following unused import(s) in contracts/mock/DealProvider.sol should be removed:
	-import "@poolzfinance/lockdeal-nft/contracts/SimpleProviders/DealProvider/DealProvider.sol"; (contracts/mock/DealProvider.sol#4)
The following unused import(s) in contracts/mock/ERC20Token.sol should be removed:
	-import "@poolzfinance/poolz-helper-v2/contracts/token/ERC20Token.sol"; (contracts/mock/ERC20Token.sol#4)
The following unused import(s) in contracts/mock/LockDealNFT.sol should be removed:
	-import "@poolzfinance/lockdeal-nft/contracts/LockDealNFT/LockDealNFT.sol"; (contracts/mock/LockDealNFT.sol#4)
The following unused import(s) in contracts/mock/LockDealProvider.sol should be removed:
	-import "@poolzfinance/lockdeal-nft/contracts/SimpleProviders/LockProvider/LockDealProvider.sol"; (contracts/mock/LockDealProvider.sol#4)
The following unused import(s) in contracts/mock/MockProvider.sol should be removed:
	-import "@poolzfinance/lockdeal-nft/contracts/mock/MockProvider.sol"; (contracts/mock/MockProvider.sol#4)
The following unused import(s) in contracts/mock/TimedProvider.sol should be removed:
	-import "@poolzfinance/lockdeal-nft/contracts/SimpleProviders/TimedDealProvider/TimedDealProvider.sol"; (contracts/mock/TimedProvider.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-imports
```

Unit tests

All unit tests are passed.



```
    ✔ should return name of contract
    ✔ should increase leftAmount after creation
    ✔ should deacrease leftAmount after lock
    ✔ should withdraw if available and disper approved
    ✔ should not withdraw if dispenser not approved
    ✔ should create lock if approved for all
    ✔ should revert double creation
    ✔ should revert invalid signer address
    ✔ should revert if sender is invalid
    ✔ should revert zero token address
    ✔ should revert invalid amount
    ✔ should emit TokensDispensed event
    ✔ should support IERC165 interface
    ✔ should support IDispenserProvider interface
    ✔ should revert if params amount greater than leftAmount
    ✔ should revert zero params amount


  16 passing (389ms)

·························································································································
|  Solidity and Network Configuration                                                                                   |
··································································|·················|···············|················|·····················
|  Solidity: 0.8.28                                               ·  Optim: true    ·  Runs: 200    ·  viaIR: true   ·     Block: 130,000,000 gas  |
··································································|·················|···············|················|·····················
|  Methods                                                                                                              |
··································································|·················|···············|················|················|····
|  Contracts / Methods                                            ·  Min            ·  Max          ·  Avg           ·  # calls       ·  usd (avg)  |
··································································|·················|···············|················|················|····
|  DispenserProvider                                              ·                                                                     |
··································································|·················|···············|················|················|····
|      createNewPool(address[],uint256[],bytes)                   ·        294,640  ·      322,940  ·       321,171  ·            16  ·          -  |
··································································|·················|···············|················|················|····
|      dispenseLock(uint256,uint256,address,(address,uint256[])[],bytes)  ·  308,152  ·  356,350  ·  345,909  ·  8  ·  -  |
··································································|·················|···············|················|················|····
|  ERC20Token                                                     ·                                                                     |
··································································|·················|···············|················|················|····
|      approve(address,uint256)                                   ·              -  ·            -  ·        45,962  ·            16  ·          -  |
··································································|·················|···············|················|················|····
|  LockDealNFT                                                    ·                                                                     |
··································································|·················|···············|················|················|····
|      setApprovalForAll(address,bool)                            ·         24,828  ·       46,740  ·        35,784  ·             4  ·          -  |
··································································|·················|···············|················|················|····
|      setApprovedContract(address,bool)                          ·         48,143  ·       48,155  ·        48,152  ·             4  ·          -  |
··································································|·················|···············|················|················|····
|  Deployments                                                                      ·                                ·  % of limit    ·             |
··································································|·················|···············|················|················|····
|  DealProvider                                                   ·              -  ·            -  ·     1,558,658  ·         1.2 %  ·          -  |
··································································|·················|···············|················|················|····
|  DispenserProvider                                              ·              -  ·            -  ·     2,536,490  ·           2 %  ·          -  |
··································································|·················|···············|················|················|····
|  ERC20Token                                                     ·              -  ·            -  ·       648,905  ·         0.5 %  ·          -  |
··································································|·················|···············|················|················|····
|  LockDealNFT                                                    ·              -  ·            -  ·     4,305,889  ·         3.3 %  ·          -  |
··································································|·················|···············|················|················|····
|  LockDealProvider                                               ·              -  ·            -  ·     1,707,187  ·         1.3 %  ·          -  |
··································································|·················|···············|················|················|····
|  MockVaultManager                                               ·              -  ·            -  ·       322,429  ·         0.2 %  ·          -  |
··································································|·················|···············|················|················|····
|  TimedDealProvider                                              ·              -  ·            -  ·     1,854,511  ·         1.4 %  ·          -  |
··································································|·················|···············|················|················|····
|  Key                                                                                                                  |
·························································································································
|  ◯  Execution gas for this method does not include intrinsic gas overhead                                             |
·························································································································
|  △  Cost was non-zero but below the precision setting for the currency display (see options)                          |
·························································································································
|  Toolchain:  hardhat                                                                                                  |
·························································································································
```

Conclusion

Arcadia identified issues that occurred at the following repositories:

- **LockDealNFT Dispenser Provider contracts at commit #92eb746e1f46b37a986a8f86b2b06ddda6adf002**

All of the issues were resolved at commit #170f9d6e66baaec9e5fadc54a572af9712bcf6da

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.