Security Audit Report

Poolz LockDealNFT.Builders & TokenNFTConnector

PREPARED FOR:

Poolz, Poolz.Finance

ARCADIA CONTACT INFO

Email: audits@arcadiamgroup.com

Telegram: https://t.me/thearcadiagroup

Revision history

Date	Reason	Commit
03/20/2024	Initial Audit Scope	**TokenNFTConnector**: #b51c1b6c989d2d555190aba93e3603198b79d1d5 **LockDealNFT.Builders**: #bce4d1bf1f58dfacaeca288a5c445f0420b95faa
05/17/2024	Review Of Remediations	**TokenNFTConnector**: #0b577e6b051aaf7c0a604b51cffbf7c5c1a66 **LockDealNFT.Builders**: #52f0ef17577183290d5d98ff4576e007fc0e5f

Table of Contents

Executive Summary

1. Introduction and Audit Scope
2. Audit Summary

Findings in Manual Audit

1. Deactivated slippage control could result in excessive slippage
2. Fee-on-transfer tokens are not supported as swap tokens
3. Unsafe transfers break compatibility with multiple ERC20 tokens
4. createLeaderboard is incompatible with custom tokens such as USDT
5. ConnectorManageable owner can front-run fees to nearly 100% to steal the users' tokens
6. Missing check may result in projectOwnerFee exceeding 100%
7. Invalid revert message
8. Missing zero address checks could lead to redeploying contracts
9. Using an older version of OpenZeppelin libraries can pose significant risks
10. Typo in comments
11. Inadequate naming could lead to confusion
12. Missing or Incomplete NatSpec
13. Missing events on crucial functions and important state changes
14. State variables only set in the constructor should be declared as immutable
15. Split revert statements
16. Loop gas usage optimization
17. Redundant computation
18. Use custom errors instead of require statements

Automated Audit

Static Analysis with Slither

Unit Tests

Tests Coverage

Disclaimer

Executive Summary

1. Introduction and Audit Scope

A Representative of Poolz ("CLIENT") engaged The Arcadia Group ("Arcadia"), a software development, research, and security company, to conduct a review of the Poolz smart contracts located in the following GitHub repositories:

- **The-Poolz/LockDealNFT.Builders at tag v1.0.0, commit #bce4d1bf1f58dfacaeca288a5c445f0420b95faa**
- **The-Poolz/TokenNFTConnector at tag v1.1.1, commit #b51c1b6c989d2d555190aba93e3603198b79d1d5**

a. Review Team

Jihed Chalghaf - Security Researcher and Engineer

Van Cam Pham - Lead Security Engineer and Secondary Review

Joel Farris - Project Manager

b. Coverage

The scope of this audit included the following files:

TokenNFTConnector/contracts/TokenNFTConnector.sol
TokenNFTConnector/contracts/ConnectorManageable.sol
TokenNFTConnector/contracts/interfaces/ISwapRouter.sol
TokenNFTConnector/contracts/interfaces/IDelayVaultProvider.sol
LockDealNFT.Builders/contracts/SimpleRefundBuilder/RefundBuilderInternal.sol
LockDealNFT.Builders/contracts/SimpleRefundBuilder/RefundBuilderState.sol
LockDealNFT.Builders/contracts/SimpleRefundBuilder/SimpleRefundBuilder.sol
LockDealNFT.Builders/contracts/SimpleBuilder/SimpleBuilder.sol
LockDealNFT.Builders/contracts/Builder/BuilderInternal.sol
LockDealNFT.Builders/contracts/Builder/BuilderModifiers.sol

LockDealNFT.Builders/contracts/Builder/BuilderState.sol

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 other 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
- Static Analysis:
    - 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 **19** issues found, **0** of which were deemed to be ‘critical’, and **1** of which were rated as ‘high’. At the end of these issues were found throughout the review of a rapidly changing codebase and not a final static point in time.

Severity Rating	Number of Original Occurrences	Number of Remaining Occurrences
CRITICAL	0	0
HIGH	1	0
MEDIUM	5	0
LOW	4	0
INFORMATIONAL	4	0
GAS	5	0

Findings in Manual Audit

1. Deactivated slippage control could result in excessive slippage

Issue ID

LDN-1

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: High, Likelihood: Medium

Code Segment

```
function createLeaderboard(
        uint256 amountIn,
        SwapParams[] calldata poolsData
    ) external whenNotPaused nonReentrant returns (uint256 amountOut)
{
        IERC20 tokenToSwap = (poolsData.length > 0)
            ? IERC20(poolsData[0].token)
            : pairToken;
        require(
            tokenToSwap.allowance(msg.sender, address(this)) ≥
amountIn,
            "TokenNFTConnector: no allowance"
        );
        tokenToSwap.transferFrom(msg.sender, address(this), amountIn);
```

```
        tokenToSwap.approve(address(swapRouter), amountIn);

        amountOut = swapRouter.exactInput(
            ISwapRouter.ExactInputParams({
                path: getBytes(poolsData),
                recipient: address(this),
                amountIn: amountIn,
                amountOutMinimum: 0
            })
        );
        amountOut = calcMinusFee(amountOut);
        require(
            !checkIncreaseTier(msg.sender, amountOut),
            "TokenNFTConnector: please update your tier level"
        );
        token.approve(address(delayVaultProvider), amountOut);
        uint256[] memory delayParams = new uint256[](1);
        delayParams[0] = amountOut;
        delayVaultProvider.createNewDelayVault(msg.sender,
delayParams);
    }
```

Description

An attacker can monitor the mempool and put two transactions before and after the user's transaction (sandwich attack). For example, when an attacker spots a large trade, executes their own trade first to manipulate the POOLX price, and then profits by selling their tokens after the user's trade is executed.

Code Location

```
TokenNFTConnector/contracts/TokenNFTConnector.sol
```

Recommendation

Allow users to specify the minimum output amount and revert the transaction if it is not satisfied.

2. Fee-on-transfer tokens are not supported as swap tokens

Issue ID

LDN-2

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Medium, Likelihood: Medium

Code Segment

```
function createLeaderboard(
        uint256 amountIn,
        SwapParams[] calldata poolsData
    ) external whenNotPaused nonReentrant returns (uint256 amountOut)
{
        IERC20 tokenToSwap = (poolsData.length > 0)
            ? IERC20(poolsData[0].token)
            : pairToken;
        require(
```

```
            tokenToSwap.allowance(msg.sender, address(this)) ≥
amountIn,
            "TokenNFTConnector: no allowance"
        );

        tokenToSwap.transferFrom(msg.sender, address(this), amountIn);
        tokenToSwap.approve(address(swapRouter), amountIn);

        amountOut = swapRouter.exactInput(
            ISwapRouter.ExactInputParams({
                path: getBytes(poolsData),
                recipient: address(this),
                amountIn: amountIn,
                amountOutMinimum: 0
            })
        );
```

Description

The highlighted `swapRouter.exactInput` call will revert if `tokenToSwap` is a fee-on-transfer token because the actual received amount will be less than `amountIn`.

Additionally, this leads to providing an unnecessarily higher approval amount during the `approve` call.

Code Location

`TokenNFTConnector/contracts/TokenNFTConnector.sol`

Recommendation

Consider measuring the actual received amount through a balance check and using it instead of `amountIn`.

3. Unsafe transfers break compatibility with multiple ERC20 tokens

Issue ID

LDN-3

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Medium, Likelihood: Medium

Code Segment

```
// at TokenNFTConnector.sol
function createLeaderboard(
        uint256 amountIn,
        SwapParams[] calldata poolsData
    ) external whenNotPaused nonReentrant returns (uint256 amountOut)
{
        IERC20 tokenToSwap = (poolsData.length > 0)
            ? IERC20(poolsData[0].token)
            : pairToken;
        require(
```

```
            tokenToSwap.allowance(msg.sender, address(this)) ≥
amountIn,
            "TokenNFTConnector: no allowance"
        );

        tokenToSwap.transferFrom(msg.sender, address(this), amountIn);
```

```
// at ConnectorManageable.sol
function withdrawFee() external onlyOwner {
        uint256 balance = token.balanceOf(address(this));
        require(balance > 0, "ConnectorManageable: balance is zero");
        token.transfer(owner(), balance);
    }
```

Description

The highlighted contracts use direct ERC20 transfers instead of using a safe transfer library, which could cause issues due to the following:

- Some tokens (e.g. ZRX) don't revert on transfers.
- Tokens that don't perform the transfer and return false are still counted as a correct transfer.

If the specified conditions are satisfied, the worst-case scenario may involve depleting the contract's accumulated fees, represented by the `token (POOLX)`:

- When `tokenToSwap` matches the `token` state variable
- And `token` does not revert on transfer failure.

Fortunately, `POOLX` automatically reverts on transfer failures, greatly reducing the likelihood of encountering the worst-case scenario.

Code Location

```
TokenNFTConnector/contracts/*.sol
```

Recommendation

Consider using OpenZeppelin's SafeERC20 library.

4. **createLeaderboard** is incompatible with custom tokens such as USDT

Issue ID

LDN-4

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Medium, Likelihood: Medium

Code Segment

```
function createLeaderboard(
        uint256 amountIn,
        SwapParams[] calldata poolsData
    ) external whenNotPaused nonReentrant returns (uint256 amountOut)
{
        IERC20 tokenToSwap = (poolsData.length > 0)
            ? IERC20(poolsData[0].token)
            : pairToken;
        require(
```

```
            tokenToSwap.allowance(msg.sender, address(this)) ≥
amountIn,
            "TokenNFTConnector: no allowance"
        );

        tokenToSwap.transferFrom(msg.sender, address(this), amountIn);
        tokenToSwap.approve(address(swapRouter), amountIn);

        amountOut = swapRouter.exactInput(
            ISwapRouter.ExactInputParams({
                path: getBytes(poolsData),
                recipient: address(this),
                amountIn: amountIn,
                amountOutMinimum: 0
            })
        );
        amountOut = calcMinusFee(amountOut);
        require(
            !checkIncreaseTier(msg.sender, amountOut),
            "TokenNFTConnector: please update your tier level"
        );

        token.approve(address(delayVaultProvider), amountOut);
```

Description

Some ERC20 tokens (like USDT) do not work when changing the allowance from an existing non-zero allowance value to protect against front-running changes of approvals.

Code Location

```
TokenNFTConnector/contracts/TokenNFTConnector.sol
```

Recommendation

Set the allowance to zero before increasing the allowance and use `safeApprove` or `safeIncreaseAllowance` from OpenZeppelin's SafeERC20.

5. **ConnectorManageable** owner can front-run fees to nearly 100% to steal the users' tokens

Issue ID

LDN-5

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Medium, Likelihood: Low

Code Segment

```
uint256 public constant MAX_FEE = 1e18; // 100%

function setProjectOwnerFee(uint256 fee) external onlyOwner {
        require(fee < MAX_FEE, "ConnectorManageable: invalid fee");
        projectOwnerFee = fee;
    }
```

Description

The contract owner possesses the authority to establish the fee rate at an excessively high `99.99%`. This poses a significant disadvantage to users, as fees should be subject to a reasonable upper limit, such as `30%`, to prevent potential griefing. For instance, if the fee is set at the maximum allowable rate `(1e18 - 1)`, a user could receive only `1 wei` instead of the expected `10e17 wei` of the output `token`.

Code Location

```
TokenNFTConnector/contracts/ConnectorManageable.sol
```

Recommendation

Consider setting a fair upper bound for the contract `fee`.

6. Missing check may result in **projectOwnerFee** exceeding 100%

Issue ID

LDN-6

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Medium, Likelihood: Low

Code Segment

```
constructor(IERC20 _token, uint256 _projectOwnerFee)
Ownable(msg.sender) {
        require(
            address(_token) ≠ address(0),
            "ConnectorManageable: ZERO_ADDRESS"
        );
        token = _token;
        projectOwnerFee = _projectOwnerFee;
    }
```

Description

During the deployment of the `ConnectorManageable` contract, the supplied fee amount isn't verified. This oversight could lead to the establishment of a fee exceeding `100%`, potentially causing an underflow within the `calcMinusFee` function when determining the token amount to be sent at the conclusion of the `createLeaderboard` function.

```
function calcMinusFee(
        uint256 amount
    ) public view returns (uint256 leftAmount) {
        leftAmount = amount - (amount * projectOwnerFee) / MAX_FEE;
    }
```

Code Location

```
TokenNFTConnector/contracts/ConnectorManageable.sol
```

Recommendation

Consider checking the initial fee amount during the contract deployment.

```
constructor(IERC20 _token, uint256 _projectOwnerFee)
Ownable(msg.sender) {
        require(
            address(_token) != address(0),
            "ConnectorManageable: ZERO_ADDRESS"
        );
        require(_projectOwnerFee < MAX_FEE, "ConnectorManageable:
invalid fee");
        token = _token;
        projectOwnerFee = _projectOwnerFee;
    }
```

7. Invalid revert message

Issue ID

LDN-7

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Low, Likelihood: Medium

Code Segment

```
function _userDataIterator(
        Rebuilder memory data
    ) internal firewallProtectedSig(0xbbc1f709) {
```

```
        uint256 length = data.userData.userPools.length;
        require(
            length > 0,
            "SimpleRefundBuilder: addressParams must contain exactly 3
addresses"
        );
```

Description

The highlighted require statement verifies if the user has supplied at least one pool data, however, it reverts with a misleading error message.

Code Location

```
LockDealNFT.Builders/contracts/SimpleRefundBuilder/RefundBuilderInte
rnal.sol
```

Recommendation

Consider changing the revert message to:

```
function _userDataIterator(
        Rebuilder memory data
    ) internal firewallProtectedSig(0xbbc1f709) {
        uint256 length = data.userData.userPools.length;
        require(
            length > 0,
            "SimpleRefundBuilder: invalid user length"
        );
```

8. Missing zero address checks could lead to redeploying contracts

Issue ID

LDN-8

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Low, Likelihood: Low

Code Segment

```
// at SimpleRefundBuilder.sol
constructor(ILockDealNFT _nft, IProvider _refund, IProvider
_collateral) {
        lockDealNFT = _nft;
        refundProvider = _refund;
        collateralProvider = _collateral;
    }
```

```
// at SimpleBuilder.sol
constructor(ILockDealNFT _nft) {
        lockDealNFT = _nft;
    }
```

Description

During the `SimpleRefundBuilder` deployment, the deployer can mistakenly provide a zero address for either `lockDealNFT`, `refundProvider` or `collateralProvider` with no way of updating them after the deployment. The same applies for `lockDealNFT` during the `SimpleBuilder` deployment. This could lead to redeploying the entire contract and incurring gas loss due to the initial deployment.

Code Location

```
LockDealNFT.Builders/contracts/SimpleRefundBuilder/SimpleRefundBuild
er.sol
LockDealNFT.Builders/contracts/SimpleBuilder/SimpleBuilder.sol
```

Recommendation

Consider adding zero address checks for all the specified state variables.

```
// at SimpleRefundBuilder.sol
constructor(ILockDealNFT _nft, IProvider _refund, IProvider
_collateral) {
        require(
            address(_nft) ≠ address(0),
            "SimpleRefundBuilder: zero address"
        );
        require(
            address(_refund) ≠ address(0),
            "SimpleRefundBuilder: zero address"
        );
        require(
            address(_collateral) ≠ address(0),
```

```
            "SimpleRefundBuilder: zero address"
        );
        lockDealNFT = _nft;
        refundProvider = _refund;
        collateralProvider = _collateral;
    }
```

```
// at SimpleBuilder.sol
constructor(ILockDealNFT _nft) {
        require(
            address(_nft) != address(0),
            "SimpleBuilder: zero address"
        );
        lockDealNFT = _nft;
    }
```

9. Using an older version of **OpenZeppelin** libraries can pose significant risks

Issue ID

LDN-9

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Low, Likelihood: Low

Code Segment

```
"dependencies": {
        "@ironblocks/firewall-consumer": "^1.0.5",
        "@openzeppelin/contracts": "^4.9.6",
        "@poolzfinance/poolz-helper-v2": "^2.3.5",
        "@poolzfinance/lockdeal-nft": "^0.8.0",
        "@poolzfinance/refund-provider": "^0.8.0",
        "@poolzfinance/collateral-provider": "^0.8.1"
    }
```

Description

Currently the protocol is using version `4.9.6` of the `OpenZeppelin` contracts which are continuously updated to eliminate any bugs and vulnerabilities.

Code Location

```
LockDealNFT.Builders/package.json
```

Recommendation

Consider using the latest version **(**`5.0.2` so far**)**.

10. Typo in comments

Issue ID

LDN-10

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Informational

Description

The following comment, which occurs twice, contains a typo:

```
// one time transfer for deacrease number transactions
```

Code Location

```
LockDealNFT.Builders/contracts/SimpleBuilder/SimpleBuilder.sol
LockDealNFT.Builders/contracts/SimpleRefundBuilder/SimpleRefundBuild
er.sol
```

Recommendation

Consider changing to:

```
"one time transfer for decreasing the number of transactions"
```

11. Inadequate naming could lead to confusion

Issue ID

LDN-11

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Informational

Code Segment

```
function onERC721Received(
        address operator,
        address user,
        uint256 poolId,
        bytes calldata data
    )
```

```
    /// @notice Creates the collateral provider
    /// @param data Rebuilder struct containing mainCoin data
    /// @param collateralFinishTime Finish time for refund
    /// @return poolId Collateral pool ID
    function _createCollateralProvider(
        Rebuilder memory data,
        uint256 collateralFinishTime
    ) internal firewallProtectedSig(0x4516d406) returns (uint256
poolId) {
```

```
    /// @notice Iterates over user data to create refund pools for
each user
```

```
    /// @param data Users data, paramsData, tokenPoolId, simple
provider address
    function _userDataIterator(
        Rebuilder memory data
    ) internal firewallProtectedSig(0xbbc1f709)
```

Description

In the functions mentioned above, the `poolId` is described as *"the ID of the Collateral NFT"*. Given the presence of multiple ID types within the codebase (simple pool ID, refund pool ID, collateral pool ID), it is advisable to rename it to `collateralPoolId`. This adjustment helps prevent confusion when interpreting the code, rather than relying solely on `NatSpec` documentation.

Moreover, the `_userDataIterator` function's name lacks clarity regarding its functionality, which involves the creation of bulk refund pools. Therefore, renaming it to something like `_buildMassPools` would enhance readability and comprehension.

Code Location

```
LockDealNFT.Builders/contracts/SimpleRefundBuilder/SimpleRefundBuild
er.sol
LockDealNFT.Builders/contracts/SimpleRefundBuilder/RefundBuilderInte
rnal.sol
```

Recommendation

Consider changing to:

```
function onERC721Received(
        address operator,
        address user,
```

```
        uint256 collateralPoolId,
        bytes calldata data
    )
```

```
function _createCollateralProvider(
        Rebuilder memory data,
        uint256 collateralFinishTime
    ) internal firewallProtectedSig(0x4516d406) returns (uint256
collateralPoolId) {
```

```
function _buildMassPools(
        Rebuilder memory data
    ) internal firewallProtectedSig(0xbbc1f709)
```

12. Missing or Incomplete **NatSpec**

Issue ID

LDN-12

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Informational

Description

The `NatSpec` documentation within the contracts was discovered to be absent or incomplete on the following occasions:

```
contracts/Builder/BuilderInternal.sol:9
BuilderInternal:_concatParams
@param amount is missing
@param params is missing
@return result is missing

contracts/Builder/BuilderInternal.sol:21
BuilderInternal:_createNewNFT
Natspec is missing

contracts/Builder/BuilderInternal.sol:34
BuilderInternal:_createFirstNFT
Natspec is missing

contracts/Builder/BuilderModifiers.sol:19
BuilderModifiers:_notZeroAmount
Natspec is missing

contracts/Builder/BuilderModifiers.sol:23
BuilderModifiers:_notZeroAddress
Natspec is missing

contracts/Builder/BuilderModifiers.sol:27
BuilderModifiers:_validParamsLength
Natspec is missing

contracts/Builder/BuilderModifiers.sol:8
BuilderModifiers:notZeroAddress
Natspec is missing

contracts/Builder/BuilderModifiers.sol:13
BuilderModifiers:validUserData
Natspec is missing

contracts/Builder/BuilderState.sol:9
BuilderState:Builder
Natspec is missing

contracts/Builder/BuilderState.sol:14
BuilderState:UserPool
Natspec is missing
```

contracts/SimpleBuilder/SimpleBuilder.sol:15
SimpleBuilder:MassPoolsLocals
Natspec is missing

contracts/SimpleRefundBuilder/RefundBuilderState.sol:18
RefundBuilderState:ParamsData
Natspec is missing

contracts/SimpleRefundBuilder/RefundBuilderState.sol:27
RefundBuilderState:Rebuilder
Natspec is missing

contracts/ConnectorManageable.sol:19
ConnectorManageable:setProjectOwnerFee
@inheritdoc is missing

contracts/ConnectorManageable.sol:24
ConnectorManageable:pause
@inheritdoc is missing

contracts/ConnectorManageable.sol:28
ConnectorManageable:unpause
@inheritdoc is missing

contracts/ConnectorManageable.sol:32
ConnectorManageable:withdrawFee
@inheritdoc is missing

contracts/ConnectorManageable.sol:38
ConnectorManageable:calcMinusFee
@inheritdoc is missing

contracts/ConnectorManageable.sol:9
ConnectorManageable:token
@inheritdoc is missing

contracts/ConnectorManageable.sol:10
ConnectorManageable:projectOwnerFee
@inheritdoc is missing

contracts/ConnectorManageable.sol:11
ConnectorManageable:MAX_FEE
@inheritdoc is missing

contracts/TokenNFTConnector.sol:45
TokenNFTConnector:createLeaderboard
@inheritdoc is missing

```
contracts/TokenNFTConnector.sol:79
TokenNFTConnector:getBytes
@inheritdoc is missing

contracts/TokenNFTConnector.sol:99
TokenNFTConnector:checkIncreaseTier
@inheritdoc is missing

contracts/TokenNFTConnector.sol:11
TokenNFTConnector:swapRouter
@inheritdoc is missing

contracts/TokenNFTConnector.sol:12
TokenNFTConnector:delayVaultProvider
@inheritdoc is missing

contracts/TokenNFTConnector.sol:13
TokenNFTConnector:pairToken
@inheritdoc is missing

contracts/TokenNFTConnector.sol:14
TokenNFTConnector:poolFee
Natspec is missing

contracts/TokenNFTConnector.sol:16
TokenNFTConnector:SwapParams
Natspec is missing

contracts/interfaces/ISwapRouter.sol:5
ISwapRouter:ExactInputParams
Natspec is missing
```

Code Location

```
LockDealNFT.Builders/contracts/Builder/*.sol
LockDealNFT.Builders/contracts/SimpleBuilder/SimpleBuilder.sol
LockDealNFT.Builders/contracts/SimpleRefundBuilder/RefundBuilderStat
e.sol
TokenNFTConnector/contracts/TokenNFTConnector.sol
TokenNFTConnector/contracts/ConnectorManageable.sol
TokenNFTConnector/contracts/interfaces/ISwapRouter.sol
```

Recommendation

Consider adding the missing [NatSpec] comments and parameters.

13. Missing events on crucial functions and important state changes

Issue ID

LDN-13

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Informational

Description

It has been observed that important functionality is missing emitting events for some functions: `ConnectorManageable.setProjectOwnerFee`, `ConnectorManageable.withdrawFee`, `TokenNFTConnector.createLeaderboard`, `SimpleBuilder.buildMassPools` and `SimpleRefundBuilder.buildMassPools`.

These functions should emit events. Events are a method of informing the transaction initiator about the actions taken by the called function.

Code Location

```
TokenNFTConnector/contracts/ConnectorManageable.sol
TokenNFTConnector/contracts/TokenNFTConnector.sol
LockDealNFT.Builders/contracts/SimpleBuilder/SimpleBuilder.sol
LockDealNFT.Builders/contracts/SimpleRefundBuilder/SimpleRefundBuild
```

```
er.sol
```

Recommendation

Make sure to emit events in all essential functions.

14. State variables only set in the constructor should be declared as **immutable**

Issue ID

LDN-14

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Gas

Code Segment

```
// at BuilderState.sol
ILockDealNFT public lockDealNFT;

// at RefundBuilderState.sol
// Instance of the refund provider contract
IProvider public refundProvider;
// Instance of the collateral provider contract
IProvider public collateralProvider;

// at TokenNFTConnector.sol
ISwapRouter public swapRouter;
```

```
IDelayVaultProvider public delayVaultProvider;
IERC20 public pairToken;
uint24 private poolFee; // last pair fee
```

Description

The `BuilderState.lockDealNFT`, `RefundBuilderState.refundProvider`, `RefundBuilderState.collateralProvider`, `TokenNFTConnector.swapRouter`, `TokenNFTConnector.delayVaultProvider`, `TokenNFTConnector.pairToken` and `TokenNFTConnector.poolFee` state variables are only assigned in the constructor. These variables can be declared as immutable to make them read-only, but only assignable in the constructor, reducing gas costs.

Code Location

```
TokenNFTConnector/contracts/TokenNFTConnector.sol
LockDealNFT.Builders/contracts/Builder/BuilderState.sol
LockDealNFT.Builders/contracts/SimpleRefundBuilder/RefundBuilderStat
e.sol
```

Recommendation

Consider marking the mentioned state variables as `immutable`.

15. Split revert statements

Issue ID

LDN-15

Status

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

Risk Level

Severity: Gas

Code Segment

```
        require(
            address(_swapRouter) ≠ address(0) &&
                address(_delayVaultProvider) ≠ address(0) &&
                address(_pairToken) ≠ address(0),
            "TokenNFTConnector: ZERO_ADDRESS"
        );
```

Description

When splitting revert statements, we assert that the validity of each statement is imperative for the function's ongoing execution.

If the initial statement proves false, the function will promptly revert, and subsequent require statements will not undergo evaluation. This measure aims to economize gas usage by bypassing the evaluation of subsequent require statements.

Code Location

```
TokenNFTConnector/contracts/TokenNFTConnector.sol
```

Recommendation

Consider changing to:

```
        require(
            address(_swapRouter) ≠ address(0),
            "TokenNFTConnector: ZERO_ADDRESS"
        );
```

```
        require(
            address(_delayVaultProvider) != address(0),
            "TokenNFTConnector: ZERO_ADDRESS"
        );
        require(
            address(_pairToken) != address(0),
            "TokenNFTConnector: ZERO_ADDRESS"
        );
```

16. Loop gas usage optimization

Issue ID

LDN-16

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Gas

Code Segment

```
// at BuilderInternal.sol
function _concatParams(
        uint amount,
        uint256[] calldata params
    ) internal pure returns (uint256[] memory result) {
        uint256 length = params.length;
        result = new uint256[](length + 1);
```

```
        result[0] = amount;

        for (uint256 i = 0; i < length; ) {
            result[i + 1] = params[i];
            unchecked {
                ++i;
            }
        }
    }
```

```
// at TokenNFTConnector.sol
function getBytes(
        SwapParams[] calldata data
    ) public view returns (bytes memory result) {
        //@audit-gas - ++i could be used in an unchecked block
        for (uint256 i; i < data.length; ++i) {
            require(
                data[i].token != address(0),
                "TokenNFTConnector: ZERO_ADDRESS"
            );
            result = abi.encodePacked(
                result,
                abi.encodePacked(data[i].token, data[i].fee)
            );
        }
        // add last path element
        result = abi.encodePacked(
            result,
            abi.encodePacked(address(pairToken), poolFee,
address(token))
        );
```

```
    }
```

Description

It was identified that the for loops employed in the contracts can be gas optimized by the following principles:

- Unnecessary reading of the array length on each iteration wastes gas.
- Loop counters do not need to be set to 0, since uint256 is already initialized to 0.
- It is also possible to further optimize loops by using unchecked loop index incrementing and decrementing.

Code Location

```
LockDealNFT.Builders/contracts/Builder/BuilderInternal.sol
TokenNFTConnector/contracts/TokenNFTConnector.sol
```

Recommendation

Consider changing to:

```
// at BuilderInternal.sol
for (uint256 i; i < length; ) {
            result[i + 1] = params[i];
            unchecked {
                ++i;
            }
        }

// at TokenNFTConnector.sol
function getBytes(
        SwapParams[] calldata data
    ) public view returns (bytes memory result) {
```

```
        uint256 length = data.length;
        for (uint256 i; i < length; ) {
            require(
                data[i].token != address(0),
                "TokenNFTConnector: ZERO_ADDRESS"
            );
            result = abi.encodePacked(
                result,
                abi.encodePacked(data[i].token, data[i].fee)
            );
            unchecked {
                ++i;
            }
        }
        // add last path element
        result = abi.encodePacked(
            result,
            abi.encodePacked(address(pairToken), poolFee,
address(token))
        );
    }
```

17. Redundant computation

Issue ID

LDN-17

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Gas

Code Segment

```
function _getParamsData(
        uint256 collateraPoolId,
        uint256 tokenAmount,
        uint256 firstAmount
    ) internal view returns (ParamsData memory paramsData) {
        // get refund pool ID
        uint256 refundPoolId = collateraPoolId - 2;
        // Ensure valid refund pool ID
        require(
            lockDealNFT.poolIdToProvider(refundPoolId) ==
refundProvider,
            "SimpleRefundBuilder: invalid refundPoolId"
        );
        paramsData.token = lockDealNFT.tokenOf(refundPoolId);
        paramsData.mainCoin = lockDealNFT.tokenOf(collateraPoolId);
        paramsData.provider = ISimpleProvider(
            address(lockDealNFT.poolIdToProvider(refundPoolId + 1))
        );
        paramsData.mainCoinAmount = tokenAmount.calcAmount(
            collateralProvider.getParams(collateraPoolId)[2]
        );
        paramsData.simpleParams = paramsData.provider.getParams(
            refundPoolId + 1
        );
        paramsData.simpleParams[0] = firstAmount;
```

```
    }
```

Description

The highlighted computation is redundant. Storing the computed result in a new variable and using it twice would save some gas.

Code Location

```
LockDealNFT.Builders/contracts/SimpleRefundBuilder/RefundBuilderState.sol
```

Recommendation

Consider changing to:

```
function _getParamsData(
        uint256 collateraPoolId,
        uint256 tokenAmount,
        uint256 firstAmount
    ) internal view returns (ParamsData memory paramsData) {
        // get refund pool ID
        uint256 refundPoolId = collateraPoolId - 2;
        // Ensure valid refund pool ID
        require(
            lockDealNFT.poolIdToProvider(refundPoolId) == refundProvider,
            "SimpleRefundBuilder: invalid refundPoolId"
        );
        paramsData.token = lockDealNFT.tokenOf(refundPoolId);
        paramsData.mainCoin = lockDealNFT.tokenOf(collateraPoolId);
        uint256 poolId = refundPoolId + 1;
        paramsData.provider = ISimpleProvider(
```

```
            address(lockDealNFT.poolIdToProvider(poolId))
        );
        paramsData.mainCoinAmount = tokenAmount.calcAmount(
            collateralProvider.getParams(collateraPoolId)[2]
        );
        paramsData.simpleParams = paramsData.provider.getParams(
            poolId
        );
        paramsData.simpleParams[0] = firstAmount;
    }
```

18. Use custom errors instead of require statements

Issue ID

LDN-18

Status

Resolved

TokenNFTConnector: #0b577e6b051aaf7c0a604b51cffbf7c5c1ac1a66

LockDealNFT.Builders: #52f0ef17577183290d5d98ff4576e007fc004e5f

Risk Level

Severity: Gas

Description

Custom errors are available from solidity version `0.8.4`. Instead of using error strings, to reduce deployment and runtime cost, you should use custom errors.

Code Location

```
TokenNFTConnector/contracts/*.sol
LockDealNFT.Builders/contracts/Builder/BuilderModifiers.sol
LockDealNFT.Builders/contracts/SimpleBuilder/SimpleBuilder.sol
LockDealNFT.Builders/contracts/SimpleRefundBuilder/*.sol
```

Recommendation

Consider replacing the require statements with `if (something) revert CustomError()` type of checks.

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.

The following shows the results found by the static analysis by Slither. We reviewed the results, and, except the issues that were identified previously, all the other issues found by Slither are false positives.

`TokenNFTConnector`

```
TokenNFTConnector.getBytes(TokenNFTConnector.SwapParams[]) (contracts/TokenNFTConnector.sol#79-97) calls abi.encodePacke
d() with multiple dynamic arguments:
        - result = abi.encodePacked(result,abi.encodePacked(data[i].token,data[i].fee)) (contracts/TokenNFTConnector.sol
#87-90)
TokenNFTConnector.getBytes(TokenNFTConnector.SwapParams[]) (contracts/TokenNFTConnector.sol#79-97) calls abi.encodePacke
d() with multiple dynamic arguments:
        - result = abi.encodePacked(result,abi.encodePacked(address(pairToken),poolFee,address(token))) (contracts/Token
NFTConnector.sol#93-96)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#abi-encodePacked-collision
INFO:Detectors:
ConnectorManageable.withdrawFee() (contracts/ConnectorManageable.sol#32-36) ignores return value by token.transfer(owner
(),balance) (contracts/ConnectorManageable.sol#35)
TokenNFTConnector.createLeaderboard(uint256,TokenNFTConnector.SwapParams[]) (contracts/TokenNFTConnector.sol#45-77) igno
res return value by tokenToSwap.transferFrom(msg.sender,address(this),amountIn) (contracts/TokenNFTConnector.sol#57)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
TokenNFTConnector.createLeaderboard(uint256,TokenNFTConnector.SwapParams[]) (contracts/TokenNFTConnector.sol#45-77) igno
res return value by tokenToSwap.approve(address(swapRouter),amountIn) (contracts/TokenNFTConnector.sol#58)
TokenNFTConnector.createLeaderboard(uint256,TokenNFTConnector.SwapParams[]) (contracts/TokenNFTConnector.sol#45-77) igno
res return value by token.approve(address(delayVaultProvider),amountOut) (contracts/TokenNFTConnector.sol#73)
TokenNFTConnector.createLeaderboard(uint256,TokenNFTConnector.SwapParams[]) (contracts/TokenNFTConnector.sol#45-77) igno
res return value by delayVaultProvider.createNewDelayVault(msg.sender,delayParams) (contracts/TokenNFTConnector.sol#76)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Pragma version^0.8.0 (contracts/ConnectorManageable.sol#2) allows old versions
Pragma version^0.8.0 (contracts/TokenNFTConnector.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/IDelayVaultProvider.sol#2) allows old versions
Pragma version^0.8.0 (contracts/interfaces/ISwapRouter.sol#2) allows old versions
solc-0.8.23 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
ConnectorManageable.token (contracts/ConnectorManageable.sol#9) should be immutable
TokenNFTConnector.delayVaultProvider (contracts/TokenNFTConnector.sol#12) should be immutable
TokenNFTConnector.pairToken (contracts/TokenNFTConnector.sol#13) should be immutable
TokenNFTConnector.poolFee (contracts/TokenNFTConnector.sol#14) should be immutable
TokenNFTConnector.swapRouter (contracts/TokenNFTConnector.sol#11) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutabl
e
```

LockDealNFT.Builders

```
SimpleRefundBuilder.onERC721Received(address,address,uint256,bytes).locals (contracts/SimpleRefundBuilder/SimpleRefundBuilder.sol#29) is a local variable never initialized
SimpleRefundBuilder.buildMassPools(address[],BuilderState.Builder,uint256[][],bytes,bytes).locals (contracts/SimpleRefundBuilder/SimpleRefundBuilder.sol#63) is a local variable never initialized
SimpleBuilder.buildMassPools(address[],BuilderState.Builder,uint256[],bytes).locals (contracts/SimpleBuilder/SimpleBuilder.sol#40) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
RefundBuilderInternal._createFirstNFT(RefundBuilderState.Rebuilder,address) (contracts/SimpleRefundBuilder/RefundBuilderInternal.sol#23-37) ignores return value by lockDealNFT.mintForProvider(data.userData.userPools[0].user,refundProvider) (contracts/SimpleRefundBuilder/RefundBuilderInternal.sol#27)
RefundBuilderInternal._updateCollateralData(RefundBuilderState.Rebuilder,address,uint256) (contracts/SimpleRefundBuilder/RefundBuilderInternal.sol#67-85) ignores return value by lockDealNFT.safeMintAndTransfer(address(this),data.paramsData.mainCoin,from,data.paramsData.mainCoinAmount,dealProvider,data.mainCoinSignature) (contracts/SimpleRefundBuilder/RefundBuilderInternal.sol#73-80)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
BuilderInternal._createNewNFT(ISimpleProvider,uint256,BuilderState.UserPool,uint256[]) (contracts/Builder/BuilderInternal.sol#21-32) has external calls inside a loop: poolId = lockDealNFT.mintForProvider(userData.user,provider) (contracts/Builder/BuilderInternal.sol#28)
BuilderInternal._createNewNFT(ISimpleProvider,uint256,BuilderState.UserPool,uint256[]) (contracts/Builder/BuilderInternal.sol#21-32) has external calls inside a loop: provider.registerPool(poolId,params) (contracts/Builder/BuilderInternal.sol#30)
BuilderInternal._createNewNFT(ISimpleProvider,uint256,BuilderState.UserPool,uint256[]) (contracts/Builder/BuilderInternal.sol#21-32) has external calls inside a loop: lockDealNFT.cloneVaultId(poolId,tokenPoolId) (contracts/Builder/BuilderInternal.sol#31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
INFO:Detectors:
Pragma version^0.8.0 (contracts/Builder/BuilderInternal.sol#2) allows old versions
Pragma version^0.8.0 (contracts/Builder/BuilderModifiers.sol#2) allows old versions
Pragma version^0.8.0 (contracts/Builder/BuilderState.sol#2) allows old versions
Pragma version^0.8.0 (contracts/SimpleBuilder/SimpleBuilder.sol#2) allows old versions
Pragma version^0.8.0 (contracts/SimpleRefundBuilder/RefundBuilderInternal.sol#2) allows old versions
Pragma version^0.8.0 (contracts/SimpleRefundBuilder/RefundBuilderState.sol#2) allows old versions
Pragma version^0.8.0 (contracts/SimpleRefundBuilder/SimpleRefundBuilder.sol#2) allows old versions
Pragma version^0.8.0 (contracts/mock/CollateralProvider.sol#2) allows old versions
Pragma version^0.8.0 (contracts/mock/DealProvider.sol#2) allows old versions
Pragma version^0.8.0 (contracts/mock/LockDealNFT.sol#2) allows old versions
Pragma version^0.8.0 (contracts/mock/LockDealProvider.sol#2) allows old versions
Pragma version^0.8.0 (contracts/mock/MockProvider.sol#2) allows old versions
Pragma version^0.8.0 (contracts/mock/MockVaultManager.sol#2) allows old versions
Pragma version^0.8.0 (contracts/mock/RefundProvider.sol#2) allows old versions
Pragma version^0.8.0 (contracts/mock/TimedProvider.sol#2) allows old versions
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
BuilderState.lockDealNFT (contracts/Builder/BuilderState.sol#7) should be immutable
RefundBuilderState.collateralProvider (contracts/SimpleRefundBuilder/RefundBuilderState.sol#16) should be immutable
RefundBuilderState.refundProvider (contracts/SimpleRefundBuilder/RefundBuilderState.sol#14) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Unit Tests

All tests ran successfully, as demonstrated below:

`TokenNFTConnector`

```
Connector Manageable
  ✓ should set owner address after creation
  ✓ should pause contract
  ✓ should unpause contract
  ✓ should set fee amount
  ✓ should revert if the fee balance is empty
  ✓ should pause createLeaderboard
  ✓ owner can't set invalid fee amount
  ✓ should return the amount after deducting fee
  ✓ withdraw fee (49ms)

TokenNFTConnector
  ✓ should return name of the contract
  ✓ should return version of the contract
  ✓ should increase delay NFT counter (40ms)
  ✓ should increase user delay amount (52ms)
  ✓ should revert if no allowance
  ✓ should revert invalid tier swap (59ms)
  ✓ should return true if the level has increased
  ✓ should return false if the level doesn't increase


17 passing (662ms)
```

`LockDealNFT.Builders`

```
onERC721Received Collateral tests
  ✓ should return Collateral NFT deposit after rebuilding (412ms)
  ✓ should update collateral data (399ms)
  ✓ should set collateral pool id to new refunds (375ms)
  ✓ should create nft for main coin transfer (362ms)
  ✓ should revert invalid nft token (127ms)
  ✓ should revert empty data
  ✓ should revert ivalid collateral pool id (69ms)
  ✓ should revert transfer not from owner
```

```
  Simple Builder tests
Gas Used: 2116456
Price per one pool: 211645
    ✓ should create 10 dealProvider pools (525ms)
Gas Used: 10238076
Price per one pool: 204761
    ✓ should create 50 dealProvider pools (2457ms)
Gas Used: 20379734
Price per one pool: 203797
    ✓ should create 100 dealProvider pools (4937ms)
Gas Used: 2414701
Price per one pool: 241470
    ✓ should create 10 lockProvider pools (647ms)
Gas Used: 11650693
Price per one pool: 233013
    ✓ should create 50 lockProvider pools (3133ms)
Gas Used: 23195826
Price per one pool: 231958
    ✓ should create 100 lockProvider pools (6181ms)
Gas Used: 2942163
Price per one pool: 294216
    ✓ should create 10 timedProvider pools (817ms)
Gas Used: 14243449
Price per one pool: 284868
    ✓ should create 50 timedProvider pools (4024ms)
Gas Used: 28370123
Price per one pool: 283701
    ✓ should create 100 timedProvider pools (8125ms)
```

```
  Simple Refund Builder tests
Gas Used: 5026483
Price per one pool: 502648
    ✓ should create 10 simple refund pools with dealProvider (1310ms)
Gas Used: 21372737
Price per one pool: 427454
    ✓ should create 50 simple refund pools with dealProvider (5952ms)
Gas Used: 41806300
Price per one pool: 418063
    ✓ should create 100 simple refund pools with dealProvider (12340ms)
Gas Used: 5318004
Price per one pool: 531800
    ✓ should create 10 simple refund pools with lockProvider (1512ms)
Gas Used: 22787368
Price per one pool: 455747
    ✓ should create 50 simple refund pools with lockProvider (6804ms)
Gas Used: 44624492
Price per one pool: 446244
    ✓ should create 100 simple refund pools with lockProvider (13800ms)
Gas Used: 5845479
Price per one pool: 584547
    ✓ should create 10 simple refund pools with timedProvider (2024ms)
Gas Used: 25380101
Price per one pool: 507602
    ✓ should create 50 simple refund pools with timedProvider (7826ms)
Gas Used: 49798792
Price per one pool: 497987
    ✓ should create 100 simple refund pools with timedProvider (16271ms)


  26 passing (2m)
```

Tests Coverage

`TokenNFTConnector`

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	70	100	100	
ConnectorManageable.sol	100	64.29	100	100	
TokenNFTConnector.sol	100	75	100	100	
contracts/interfaces/	100	100	100	100	
IDelayVaultProvider.sol	100	100	100	100	
ISwapRouter.sol	100	100	100	100	
contracts/mocks/	100	66.67	100	100	
DelayMock.sol	100	75	100	100	
ERC20Token.sol	100	100	100	100	
SwapperMock.sol	100	50	100	100	
All files	100	69.44	100	100	

`LockDealNFT.Builders`

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Builder/	100	50	100	100	
BuilderInternal.sol	100	50	100	100	
BuilderModifiers.sol	100	50	100	100	
BuilderState.sol	100	100	100	100	
SimpleBuilder/	100	50	100	100	
SimpleBuilder.sol	100	50	100	100	
SimpleRefundBuilder/	100	58.33	100	100	
RefundBuilderInternal.sol	100	50	100	100	
RefundBuilderState.sol	100	50	100	100	
SimpleRefundBuilder.sol	100	72.22	100	100	
mock/	100	100	100	100	
CollateralProvider.sol	100	100	100	100	
DealProvider.sol	100	100	100	100	
LockDealNFT.sol	100	100	100	100	
LockDealProvider.sol	100	100	100	100	
MockProvider.sol	100	100	100	100	
MockVaultManager.sol	100	100	100	100	
RefundProvider.sol	100	100	100	100	
TimedProvider.sol	100	100	100	100	
All files	100	56.06	100	100	

In both repositories, the unit tests failed to address revert cases, as illustrated below:

ConnectorManageable.sol

```
constructor(IERC20 _token, uint256 _projectOwnerFee) Ownable(msg.sender) {
    E require(address(_token) != address(0),"ConnectorManageable: ZERO_ADDRESS");
    token = _token;
    projectOwnerFee = _projectOwnerFee;
}

function setProjectOwnerFee(uint256 fee) external E onlyOwner {
    require(fee < MAX_FEE, "ConnectorManageable: invalid fee");
    projectOwnerFee = fee;
}

function pause() external E onlyOwner {
    _pause();
}

function unpause() external E onlyOwner {
    _unpause();
}

function withdrawFee() external E onlyOwner {
    uint256 balance = token.balanceOf(address(this));
    require(balance > 0, "ConnectorManageable: balance is zero");
    token.transfer(owner(), balance);
}
```

TokenNFTConnector.sol

```
constructor(
    IERC20 _token,
    IERC20 _pairToken,
    ISwapRouter _swapRouter,
    IDelayVaultProvider _delayVaultProvider,
    uint24 _poolFee,
    uint256 _projectOwnerFee
)
    ConnectorManageable(_token, _projectOwnerFee)
    Nameable("TokenNFTConnector", "1.2.0")
{
    E require(
        address(_swapRouter) != address(0) &&
            address(_delayVaultProvider) != address(0) &&
            address(_pairToken) != address(0),
        "TokenNFTConnector: ZERO_ADDRESS"
    );
    E require(token != _pairToken, "TokenNFTConnector: SAME_TOKENS_IN_PAIR");
    swapRouter = _swapRouter;
    delayVaultProvider = _delayVaultProvider;
    pairToken = _pairToken;
    poolFee = _poolFee;
}

function createLeaderboard(
    uint256 amountIn,
    SwapParams[] calldata poolsData
) external whenNotPaused E nonReentrant returns (uint256 amountOut) {
    IERC20 tokenToSwap = (poolsData.length > 0)
        ? IERC20(poolsData[0].token)
        : pairToken;
    require(
        tokenToSwap.allowance(msg.sender, address(this)) >= amountIn,
        "TokenNFTConnector: no allowance"
    );
```

```
function getBytes(
    SwapParams[] calldata data
) public view returns (bytes memory result) {
    for (uint256 i; i < data.length; ++i) {
        E require(
            data[i].token != address(0),
            "TokenNFTConnector: ZERO_ADDRESS"
        );
        result = abi.encodePacked(
            result,
            abi.encodePacked(data[i].token, data[i].fee)
        );
    }
    // add last path element
    result = abi.encodePacked(
        result,
        abi.encodePacked(address(pairToken), poolFee, address(token))
    );
}
```

BuilderInternal.sol

```
function _createNewNFT(
    ISimpleProvider provider,
    uint256 tokenPoolId,
    UserPool memory userData,
    uint256[] memory params
) internal virtual E validUserData(userData) returns (uint256 amount) {
    amount = userData.amount;
    uint256 poolId = lockDealNFT.mintForProvider(userData.user, provider);
    params[0] = userData.amount;
    provider.registerPool(poolId, params);
    lockDealNFT.cloneVaultId(poolId, tokenPoolId);
}

function _createFirstNFT(
    ISimpleProvider provider,
    address token,
    address owner,
    uint256 totalAmount,
    uint256[] memory params,
    bytes calldata signature
) internal virtual E notZeroAddress(owner) returns (uint256 poolId) {
    poolId = lockDealNFT.safeMintAndTransfer(owner, token, msg.sender, totalAmount, provider, signature);
    provider.registerPool(poolId, params);
}
```

BuilderModifiers.sol

```
function _notZeroAmount(uint256 amount) internal pure {
    E require(amount > 0, "amount must be greater than 0");
}

function _notZeroAddress(address _address) internal pure {
    E require(_address != address(0x0), "Zero Address is not allowed");
}

function _validParamsLength(uint256 paramsLength, uint256 minLength) internal pure {
    E require(paramsLength >= minLength, "invalid params length");
}
```

SimpleBuilder.sol

```
function buildMassPools(
    address[] calldata addressParams,
    Builder calldata userData,
    uint256[] calldata params,
    bytes calldata signature
) external E firewallProtected E notZeroAddress(addressParams[1]) {
    _validParamsLength(addressParams.length, 2);
    E require(
        ERC165Checker.supportsInterface(addressParams[0], type(ISimpleProvider).interfaceId),
        "invalid provider type"
    );
    E require(userData.userPools.length > 0, "invalid user length");
```

RefundBuilderInternal.sol

```
function _createFirstNFT(
    Rebuilder memory data
) internal E firewallProtectedSig(0x3da709b8) returns (uint256 tokenPoolId){
    tokenPoolId = _createFirstNFT(data, msg.sender);
}

/// @notice Creates the first NFT for the refund provider with specified sender
/// @param data Rebuilder struct containing token data
/// @param from Address of the sender
/// @return tokenPoolId Token pool ID of the created simple NFT
function _createFirstNFT(
    Rebuilder memory data,
    address from
) internal E firewallProtectedSig(0x3da709b8) returns (uint256 tokenPoolId){
    lockDealNFT.mintForProvider(data.userData.userPools[0].user, refundProvider);
    tokenPoolId = lockDealNFT.safeMintAndTransfer(
        address(refundProvider),
        data.paramsData.token,
        from,
        data.userData.totalAmount,
        data.paramsData.provider,
        data.tokenSignature
    );
    data.paramsData.provider.registerPool(tokenPoolId, data.paramsData.simpleParams);
}
```

```
function _createCollateralProvider(
    Rebuilder memory data,
    uint256 collateralFinishTime
) internal E firewallProtectedSig(0x4516d406) returns (uint256 poolId) {
    poolId = lockDealNFT.safeMintAndTransfer(
        msg.sender,
        data.paramsData.mainCoin,
        msg.sender,
        data.paramsData.mainCoinAmount,
        collateralProvider,
        data.mainCoinSignature
    );
    uint256[] memory collateralParams = new uint256[](3);
    collateralParams[0] = data.userData.totalAmount;
    collateralParams[1] = data.paramsData.mainCoinAmount;
    collateralParams[2] = collateralFinishTime;
    collateralProvider.registerPool(poolId, collateralParams);
    lockDealNFT.cloneVaultId(poolId + 2, data.tokenPoolId);
}
```

```
function _updateCollateralData(
    Rebuilder memory data,
    address from,
    uint256 subPoolId
) internal E firewallProtectedSig(0x54c3ed4d) {
    IProvider dealProvider = lockDealNFT.poolIdToProvider(subPoolId);
    lockDealNFT.safeMintAndTransfer(
        address(this),
        data.paramsData.mainCoin,
        from,
        data.paramsData.mainCoinAmount,
        dealProvider,
        data.mainCoinSignature
    );
    // update sub collateral pool (mainCoinHolder pool)
    uint256[] memory subParams = dealProvider.getParams(subPoolId);
    subParams[0] += data.paramsData.mainCoinAmount;
    dealProvider.registerPool(subPoolId, subParams);
}
```

```
function _finalizeFirstNFT(
    Rebuilder memory data,
    uint256 collateralFinishTime
) internal E firewallProtectedSig(0xcfc2dc78) returns (uint256[] memory refundParams) {
    refundParams = _registerRefundProvider(
    data.tokenPoolId - 1,
    _createCollateralProvider(data, collateralFinishTime));
}

/// @notice Registers the refund provider
/// @param refundPoolId Refund pool ID
/// @param collateralPoolId Collateral pool ID
/// @return refundParams Refund parameter poolIdToCollateralId
function _registerRefundProvider(uint256 refundPoolId, uint256 collateralPoolId)
    internal
    E firewallProtectedSig(0x12ff3884)
    returns (uint256[] memory refundParams)
{
    refundParams = new uint256[](1);
    refundParams[0] = collateralPoolId;
    refundProvider.registerPool(refundPoolId, refundParams);
}
```

```
function _userDataIterator(
    Rebuilder memory data
) internal E firewallProtectedSig(0xbbc1f709) {
    uint256 length = data.userData.userPools.length;
    E require(length > 0, "SimpleRefundBuilder: addressParams must contain exactly 3 addresses");
```

RefundBuilderState.sol

```
function _validateParamsData(
    address[] calldata addressParams,
    uint256[][] calldata params
) internal view returns (ParamsData memory paramsData) {
    E require(addressParams.length == 3, "SimpleRefundBuilder: addressParams must contain exactly 3 addresses");
    E require(params.length == 2, "SimpleRefundBuilder: params must contain exactly 2 elements");
    E require(
        ERC165Checker.supportsInterface(addressParams[0], type(ISimpleProvider).interfaceId),
        "SimpleRefundBuilder: provider must be ISimpleProvider"
    );
    E require(addressParams[0] != address(0), "SimpleRefundBuilder: invalid provider address");
    E require(addressParams[1] != address(0), "SimpleRefundBuilder: invalid token address");
    E require(addressParams[2] != address(0), "SimpleRefundBuilder: invalid mainCoin address");
```

```
function _getParamsData(uint256 collateraPoolId, uint256 tokenAmount, uint256 firstAmount)
    internal
    view
    returns (ParamsData memory paramsData)
{
    // get refund pool ID
    uint256 refundPoolId = collateraPoolId - 2;
    // Ensure valid refund pool ID
    E require(lockDealNFT.poolIdToProvider(refundPoolId) == refundProvider, "SimpleRefundBuilder: invalid refundPoolId");
```

SimpleRefundBuilder.sol

```
function onERC721Received(address operator, address user, uint256 poolId, bytes calldata data) external virtual override E firewallProtected returns (bytes4) {
    require(msg.sender == address(lockDealNFT), "SimpleRefundBuilder: Only LockDealNFT contract allowed");
    if (operator != address(this)) {
        require(lockDealNFT.poolIdToProvider(poolId) == collateralProvider, "SimpleRefundBuilder: Invalid collateral provider");
        require(data.length > 0, "SimpleRefundBuilder: Invalid data length");
        Rebuilder memory locals;
        (
            locals.tokenSignature,
            locals.mainCoinSignature,
            locals.userData
        ) = abi.decode(data, (bytes, bytes, Builder));
        E require(locals.userData.userPools.length > 0, "SimpleRefundBuilder: invalid user length");
```

```
function buildMassPools(
    address[] calldata addressParams,
    Builder calldata userData,
    uint256[][] calldata params,
    bytes calldata tokenSignature,
    bytes calldata mainCoinSignature
) external E firewallProtected {
    Rebuilder memory locals;
    locals.paramsData = _validateParamsData(addressParams, params);
    E require(userData.userPools.length > 0, "SimpleRefundBuilder: invalid user length");
    locals.userData = userData;
    locals.tokenSignature = tokenSignature;
    locals.mainCoinSignature = mainCoinSignature;
    E require(locals.userData.totalAmount > 0, "SimpleRefundBuilder: invalid totalAmount");
```

The contract's test coverage is insufficient. Therefore, it is strongly advised to develop the missing test cases to ensure that functions revert correctly.

Issue ID

LDN-19

Risk Level

Severity: Low, Likelihood: Medium

Description

Missing unit tests to cover revert cases.

Code Location

```
TokenNFTConnector/contracts/ConnectorManageable.sol
TokenNFTConnector/contracts/TokenNFTConnector.sol
LockDealNFT.Builders/Builder/BuilderInternal.sol
LockDealNFT.Builders/Builder/BuilderModifiers.sol
LockDealNFT.Builders/SimpleBuilder/SimpleBuilder.sol
LockDealNFT.Builders/SimpleRefundBuilder/RefundBuilderInternal.sol
LockDealNFT.Builders/SimpleRefundBuilder/RefundBuilderModifiers.sol
LockDealNFT.Builders/SimpleRefundBuilder/SimpleRefundBuilder.sol
```

Recommendation

Write additional test-cases to cover the uncovered code.

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, Arcadia 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.