Security Audit Report

Hourglass Foundation

February 9th, 2023

Revision: Apr 22, 2023

PREPARED FOR:

Hourglass Foundation

ARCADIA CONTACT INFO
Email: audits@arcadiamgroup.com

Telegram: https://t.me/thearcadiagroup

Table of Contents

Executive Summary
- Introduction
- Review Team
- Project Background
- Coverage
- Methodology
- Summary
  - (HG-1) Incorrect value in event emission
    - Commit: 241134b70bcaed7a66a8d5cd03b0b5730cc4bb4f
    - Status
    - Risk level
    - Code Segment
    - Description
    - Code Location
    - Proof of Concept
    - Recommendation
    - Remediations
  - (HG-2) No implementation for function withdrawUnlocked
    - Commit: 074648cdf715000e32685f00faa6fce4cd690856
    - Status
    - Risk Level
    - Code Segment
    - Description
    - Code Location
    - Proof of concept
    - Recommendation
    - Remediations
- (HG-3) Token transfer should check if there is tax
    - Commit: ae1f97b8fb6f8b88827f3e09784f5160f29bb2f2
    - Status
    - Risk Level
    - Code Segment
    - Description
    - Code Location
    - Proof of Concept
    - Recommendation

(HG-4) Gas Usage
- Status
- Risk Level
- Code Segment
- Description
- Code Location
- Proof of Concept
- Recommendation
- Remediations

(HG-5) Unused Variables
- Status
- Risk Level
- Code Segment
- Description
- Code Location
- Proof of Concept
- Recommendation

(HG-6) Reentrancy Guards
- Status
- Risk Level
- Code Segment
- Description
- Code Location
- Proof of Concept
- Recommendation

(HG-7) Safe checks when withdraw matured to user
- Status
- Risk Level
- Commit: ae2de2739c2211b5dcb65c146b86d1ebceb2c89b
- Code Segment
- Description
- Code location
- Proof of concept
- Recommendation

(HG-8) Safe checks with total fees returned from external contract
- Status
- Risk Level
- Commit: ae2de2739c2211b5dcb65c146b86d1ebceb2c89b
- Code Segment
- Description

        - Code Location
        - Proof of concept
        - Recommendation
    - (HG-9) Add safe constructor with initializer for contracts using UUPS proxy
        - Status
        - Risk Level
        - Commit: ae2de2739c2211b5dcb65c146b86d1ebceb2c89b
        - Code Segment
        - Description
        - Code Location
        - Proof of Concept
        - Recommendation
- Automated Audit
    - Static Analysis with Slither
    - Unit Test Coverage
    - Fuzzing
- Conclusion
- Disclaimer

Executive Summary

Introduction

Hourglass Foundation engaged Arcadia to perform a security audit of their smart contracts within the hourglass-platform repository within the Pitch Foundation organization. The audit has been made by reviewing the repository at multiple commit hashes as the code changed multiple times during the audit.

Review Team

- Van Cam Pham, PhD - Security Researcher and Engineer
- Ryuhei Matsuda - Security Researcher and Engineer
- Rasikh Morani

Project Background

Hourglass Foundation is a protocol that allows for the trading of semi-fungible time-locked digital assets. Hourglass builds off of gauges of various incentivized locked stake systems like Convex's gauges.

Hourglass has a centralized bridge to transfer claims between L1s and L2s which is a pivotal part of their platform.

Coverage

For this audit, we performed research, test coverage, investigation, and review of Hourglass followed by issue reporting, along with mitigation and remediation instructions as outlined in this report. The following code repositories are considered in scope for the review. As the code was a work in progress during the start of the engagement, there are multiple states of contracts in scope.

Contracts
ethereum/eth-vaults/convex-frax-asset/ConvexFraxMaturedHoldings.sol
ethereum/eth-vaults/convex-frax-asset/ConvexFraxVault.sol
ethereum/eth-vaults/convex-frax-asset/ERC1155Receipt.sol
ethereum/eth-vaults/SeedSource.sol
ethereum/HourglassCustodian.sol
ethereum/RewardsDistributor.sol

Contracts
ethereum/eth-vaults/convex-frax-asset/ConvexFraxMaturedHoldings.sol
ethereum/eth-vaults/convex-frax-asset/ConvexFraxVault.sol
ethereum/eth-vaults/convex-frax-asset/ERC1155Receipt.sol
ethereum/eth-vaults/ERC20Intermediary.sol
ethereum/HourglassCustodian.sol
ethereum/RewardsDistributor.sol
ethereum/FeeManager.sol

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
- Static Analysis:
    - Scanning for vulnerabilities in the smart contracts using Static Code Analysis Software
    - Making a static analysis of the smart contracts using Slither
- Fuzzing
    - Arcadia assisted in writing and ensuring full coverage of fuzzing implementations

* 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.

Summary

There were **9** issues found, **1** of which were deemed to be ‘critical’, and **2** 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	1	0
HIGH	2	0
MEDIUM	3	0
LOW	1	0
INFORMATIONAL	2	0

(HG-1) Incorrect value in event emission

Commit: 241134b70bcaed7a66a8d5cd03b0b5730cc4bb4f

Status

Resolved

Risk level

Severity: Low, Likelihood: Medium

Code Segment

```
function getSeed(uint256 assetId, uint256 maturity, address asset, address
recipient, uint256 amount) external {
       /// check that we have enough asset
       require(seedBalance[asset] >= amount, "SeedSource: not enough
asset");

       // check that the caller is a vault
       address vaultCheck =
ICustodian(custodian).assetIdToMaturityToVault(assetId, maturity);
       require(vaultCheck == msg.sender && vaultCheck == recipient,
"SeedSource: caller is not vault");

       /// transfer asset to vault
       seedBalance[asset] -= amount;
       IERC20(asset).safeTransfer(recipient, amount);

       emit SeedTransferred(asset, seedBalance[asset]);
   }
```

Description

The event `SeedTransferred` should be emitted with `amount` as the second parameter.

Code Location

```
contracts/SeedSource.sol
```

Proof of Concept

–

Recommendation

Modify the code to emit the event with the second parameter as `amount`.

Remediations

As confirmed within the code and by the development team, `SeedSource` is no longer used in the latest commit.

(HG-2) No implementation for function withdrawUnlocked

Commit: **074648cdf715000e32685f00faa6fce4cd690856**

Status

Resolved

Risk Level

Severity: Critical, Likelihood: High

Code Segment

```
withdrawnAmount = IProxyVault(vault).withdrawUnlocked(
            matureVault,
            toUser,
            user,
            userAmount
        );
```

Description

The function withdrawUnlocked calls the function withdrawUnlocked of the vault contract to withdraw stake from the vault, send back to the user, and transfer the remaining to the corresponding matured holdings vault. However, there is no implementation of the function `withdrawUnlocked` in the `ConvexFraxVault` contract. This can lead to critical issues where assets will not be releasable from the vaults as the function call will always fail.

Code Location

```
contracts/HourglassCustodian.sol
```

Proof of concept

–

Recommendation

Implement the required function in the vault contract `ConvexFraxVault`

Remediations

The issue has been fixed at commit hash **d94d693e78acc8640ac972d764afc6ed63fee8d8** at which the above code segment was changed to call the implemented `withdrawMatured` function of the vault contract `ConvexFraxVault`

(HG-3) Token transfer should check if there is tax

Commit: ae1f97b8fb6f8b88827f3e09784f5160f29bb2f2

Status

Resolved

Risk Level

Severity: Medium, Likelihood: Medium

Code Segment

```
// file HourglassCustodian.sol
IERC20(assetIds[_assetId].depositToken).safeTransferFrom(
           msg.sender,
           address(this),
           100 * _maturities.length// wei
       );
// file HourglassCustodian.sol
// pull in the deposit asset from the user

IERC20(assetIds[assetId].depositToken).safeTransferFrom(msg.sender,
address(this), amount);
// file RewardsDistributor.sol
IERC20Upgradeable(_tokens[i]).safeTransferFrom(msg.sender, address(this),
_amounts[i]);
// file SeedSource.sol

IERC20(token).safeTransferFrom(msg.sender, address(this), amount);
```

Description

The safeTransferFrom function to transfer ERC20 token is called multiple times in the source code without verifying whether the being transferred token contract has tax. If the being transferred token has a tax on transfer, the amounts validation in the contract would result in an incorrect state as the amount of token transferred could be different from the actual amount of the token transferred.

Code Location

```
contracts/HourglassCustodian.sol
```

Proof of Concept

–

Recommendation

There are two methods to resolve this issue:

- Check the difference of the balances before and transfer to obtain the exact amount of the token transferred
- Make sure that the supported tokens are tax-free on transfers.

(HG-4) Gas Usage

Status

Resolved

Risk Level

Severity: Informational, Likelihood: Low

Code Segment

```
// file ConvexFraxVault.sol
     for (uint256 i; i < rewardTokens.length; i++) {
           uint256 bal = IERC20(rewardTokens[i]).balanceOf(address(this));
           if (bal > 0) {
               // if there's yields sitting here but not claimed, add it
here
               amountEarned[i] +=
IERC20(rewardTokens[i]).balanceOf(address(this));
               // todo? additional reward tokens handling from vault
(claims from wrapper, etc)
           }
       }
```

Description

Re-reading the reward token balance of the contract is not necessary.

Code Location

```
ConvexFraxVault.sol
```

Proof of Concept

–

Recommendation

Update the code `amountEarned[i] += IERC20(rewardTokens[i]).balanceOf(address(this))` to `amountEarned[i] += bal` as `bal` is set as the token balance of the contract as the statement above.

Remediations

The issue is resolved at commit hash ae1f97b8fb6f8b88827f3e09784f5160f29bb2f2

(HG-5) Unused Variables

Status

Resolved

Risk Level

Severity: Informational, Likelihood: Low

Code Segment

```
// file ConvexFraxVault.sol
function initialize(
       address[] calldata _rewardTokens, // todo probably not needed?
       address _rewardDestination,
       address _depositToken, // curve LP token
       uint256 _maturityTimestamp,
       uint256 _assetId,
       bytes calldata,
       address _deployer
   )
```

Description

The input parameters _rewardTokens, _assetId, _deployer are unused. It is recommended to either remove or comment out the variable name to silence this warning.

Code Location

```
ConvexFraxVault.sol
```

Proof of Concept

–

Recommendation

It is recommended to either remove or comment out the variable name to silence this warning.

(HG-6) Reentrancy Guards

Status

Resolved

Risk Level

Severity: Medium, Likelihood: Medium

Code Segment

```
function checkpointClaimPeriod(
       address[] memory _tokens,
       uint256[] memory _amounts,
       bytes32[] memory _merkleRoots
   ) public onlyOwner {
       for (uint256 i; i < _tokens.length; i++) {
           if (merkleRoot[_tokens[i]] != 0) revert ClaimsMustBeFrozen();

           // Increment the claim period
           claimPeriod[_tokens[i]] += 1;

           // Pull Rewards
           IERC20Upgradeable(_tokens[i]).safeTransferFrom(msg.sender,
address(this), _amounts[i]);
```

```
            // Process Fees
            uint256 fee = (_amounts[i] * platformFee) / DENOMINATOR;
            IERC20Upgradeable(_tokens[i]).safeTransfer(feeAddress, fee);

            // Update merkle root
            merkleRoot[_tokens[i]] = _merkleRoots[i];

            emit MerkleRootUpdated(_tokens[i], claimPeriod[_tokens[i]],
_merkleRoots[i]);
        }
    }
```

Description

It is recommended to have a reentrancy guard for the function

- `RewardsDistributor:checkpointClaimPeriod`
- `HourglassCustodian:upgradeMatureHoldingVault`

Code Location

```
RewardsDistributor.sol
HourglassCustodian.sol
```

Proof of Concept

–

Recommendation

Add reentrancy guards to the above functions

(HG-7) Safe checks when withdraw matured to user

Status

Resolved

Risk Level

Severity: High, Likelihood: Medium

Commit: **ae2de2739c2211b5dcb65c146b86d1ebceb2c89b**

Code Segment

```
// get the balance of the deposit token now held here
       totalWithdrawn = IERC20(depositToken).balanceOf(address(this));

       // if user triggered this withdrawal, send their portion to them &
the rest to the mature vault
       if (toUser) {
           // send to user
           IERC20(depositToken).safeTransfer(user, userAmount);
           // send to destination mature holding vault
           IERC20(depositToken).safeTransfer(destination, totalWithdrawn -
userAmount);
       } else {
           // send to destination mature holding vault
           IERC20(depositToken).safeTransfer(destination, totalWithdrawn);
       }
```

Description

Function can be reverted when transferring token to user due to

- The contract token balance `totalWithdrawn` is less than the input parameter `userAmount`

Code location

```
ConvexFraxVault.sol
```

Proof of concept

–

Recommendation

Add safe checks for checking whether the contract balance `totalWithdrawn` is greater or equal to `userAmount.`

(HG-8) Safe checks with total fees returned from external contract

Status

Resolved

Risk Level

Severity: Medium, Likelihood: Low

Commit: ae2de2739c2211b5dcb65c146b86d1ebceb2c89b

Code Segment

```
// file ConvexFraxVault.sol, function earned()
if (rewardTokens[i] == FXS) {
               // if FXS, deduct the convex fee
               amountEarned[i] = amountEarned[i] - (amountEarned[i] *
IConvexFeeRegistry(convexFeeRegistry).totalFees() / FEE_DENOMINATOR);
           }
```

Description

Function can be reverted if the function `totalFees` of the external contract `convexFeeRegistry` returns a number greater than `FEE_DENOMINATOR`.

Code Location

```
ConvexFraxVault.sol:earned
```

Proof of concept

–

Recommendation

Add safe checks for checking whether the contract balance `totalFees` is greater or equal to `FEE_DENOMINATOR.`

(HG-9) Add safe constructor with initializer for contracts using UUPS proxy

Status

Resolved

Risk Level

Severity: High, Likelihood: Medium

Commit: **ae2de2739c2211b5dcb65c146b86d1ebceb2c89b**

Code Segment

```
// file HourglassCustodian.sol
// file RewardsDistributor.sol
```

Description

Contracts that use UUPS proxy should have a default constructor to initialize the implementation's initializer. This is discussed on this thread and this.

Code Location

```
RewardsDistributor.sol
HourglassCustodian.sol
```

Proof of Concept

–

Recommendation

Add the following default constructor to the above contracts.

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() initializer {}
```

THE ARCADIA GROUP

Automated Tests and Tooling

Static Analysis with Slither

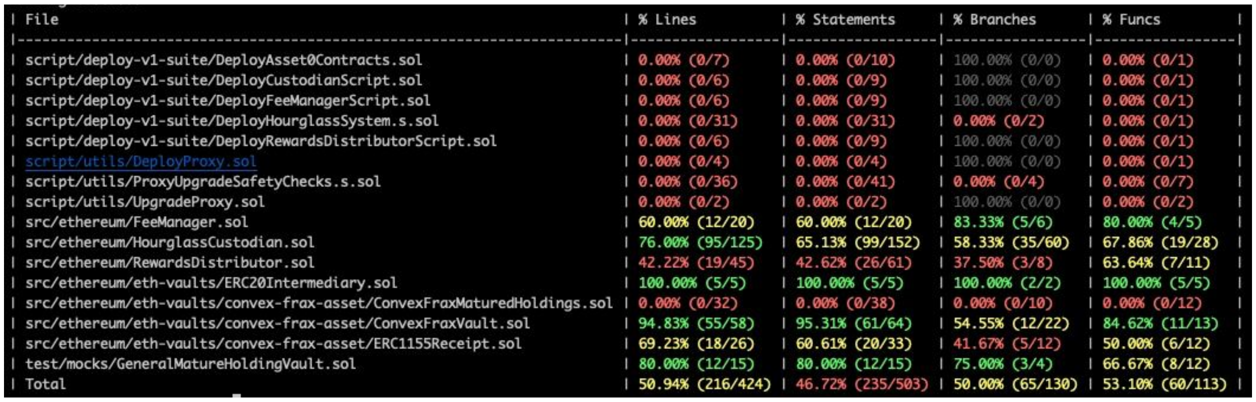
As a part of our engagement with Hourglass, we ran a static analysis against the source code using Slither, which is a Solidity static analysis framework written in Python. Slither runs a suite of vulnerability detectors and prints visual information about contract details. Slither enables developers to find vulnerabilities, enhance their code comprehension, and quickly prototype custom analyses.

While Slither is not the primary element of Arcadia's offering, in some cases, it can be useful. The following shows the results found by the static analysis by Slither. We reviewed the results, and all of the issues found by Slither are false positives. For the sake of completeness, the output has been provided below.

```
RewardsDistributor.checkpointClaimPeriod(address[],uint256[],bytes32[]).i (src/ethereum/RewardsDistributor.sol#149) is
a local variable never initialized
RewardsDistributor.updateMerkleRoots(address[],bytes32[]).i (src/ethereum/RewardsDistributor.sol#126) is a local
variable never initialized
RewardsDistributor.claimMulti(address,RewardsDistributor.ClaimData[]).i (src/ethereum/RewardsDistributor.sol#95) is a
local variable never initialized
HourglassDepositReceipt.checkpointTokenIds(uint256[]).i
(src/ethereum/eth-vaults/convex-frax-asset/ERC1155Receipt.sol#104) is a local variable never initialized
HourglassCustodian.depositMany(uint256[],uint256[],uint256[]).i (src/ethereum/HourglassCustodian.sol#368) is a local
variable never initialized
HourglassCustodian.releaseLocks(uint256,uint256[]).i (src/ethereum/HourglassCustodian.sol#560) is a local variable
never initialized
ConvexFraxVault.claimRewards().i_scope_0 (src/ethereum/eth-vaults/convex-frax-asset/ConvexFraxVault.sol#138) is a
local variable never initialized
HourglassTestingDeploymentScript.run().i (script/PitchBridgeCustodian.s.sol#144) is a local variable never initialized
ConvexFraxVault.claimRewards().i (src/ethereum/eth-vaults/convex-frax-asset/ConvexFraxVault.sol#129) is a local
variable never initialized
HourglassDepositReceipt.earned(uint256[]).i (src/ethereum/eth-vaults/convex-frax-asset/ERC1155Receipt.sol#46) is a
local variable never initialized
RewardsDistributor.freezeClaiming(address[]).i (src/ethereum/RewardsDistributor.sol#119) is a local variable never
initialized
HourglassCustodian.deployCustomTranche(uint256,uint256[]).i (src/ethereum/HourglassCustodian.sol#282) is a local
variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

Unit Test Coverage

At commit hash 59ff2b0c5841843ea6cae09927bd24d5fbb8e2a3, full unit test coverage was written by the Hourglass Foundation team.



```
| File                                                               | % Lines           | % Statements      | % Branches        | % Funcs          |
|--------------------------------------------------------------------|-------------------|-------------------|-------------------|------------------|
| script/deploy-v1-suite/DeployAsset0Contracts.sol                   | 0.00% (0/7)       | 0.00% (0/10)      | 100.00% (0/0)     | 0.00% (0/1)      |
| script/deploy-v1-suite/DeployCustodianScript.sol                   | 0.00% (0/6)       | 0.00% (0/9)       | 100.00% (0/0)     | 0.00% (0/1)      |
| script/deploy-v1-suite/DeployFeeManagerScript.sol                  | 0.00% (0/6)       | 0.00% (0/9)       | 100.00% (0/0)     | 0.00% (0/1)      |
| script/deploy-v1-suite/DeployHourglassSystem.s.sol                 | 0.00% (0/31)      | 0.00% (0/31)      | 0.00% (0/2)       | 0.00% (0/1)      |
| script/deploy-v1-suite/DeployRewardsDistributorScript.sol          | 0.00% (0/6)       | 0.00% (0/9)       | 100.00% (0/0)     | 0.00% (0/1)      |
| script/utils/DeployProxy.sol                                       | 0.00% (0/4)       | 0.00% (0/4)       | 100.00% (0/0)     | 0.00% (0/1)      |
| script/utils/ProxyUpgradeSafetyChecks.s.sol                        | 0.00% (0/36)      | 0.00% (0/41)      | 0.00% (0/4)       | 0.00% (0/7)      |
| script/utils/UpgradeProxy.sol                                      | 0.00% (0/2)       | 0.00% (0/2)       | 100.00% (0/0)     | 0.00% (0/2)      |
| src/ethereum/FeeManager.sol                                        | 60.00% (12/20)    | 60.00% (12/20)    | 83.33% (5/6)      | 80.00% (4/5)     |
| src/ethereum/HourglassCustodian.sol                                | 76.00% (95/125)   | 65.13% (99/152)   | 58.33% (35/60)    | 67.86% (19/28)   |
| src/ethereum/RewardsDistributor.sol                                | 42.22% (19/45)    | 42.62% (26/61)    | 37.50% (3/8)      | 63.64% (7/11)    |
| src/ethereum/eth-vaults/ERC20Intermediary.sol                      | 100.00% (5/5)     | 100.00% (5/5)     | 100.00% (2/2)     | 100.00% (5/5)    |
| src/ethereum/eth-vaults/convex-frax-asset/ConvexFraxMaturedHoldings.sol | 0.00% (0/32) | 0.00% (0/38)      | 0.00% (0/10)      | 0.00% (0/12)     |
| src/ethereum/eth-vaults/convex-frax-asset/ConvexFraxVault.sol      | 94.83% (55/58)    | 95.31% (61/64)    | 54.55% (12/22)    | 84.62% (11/13)   |
| src/ethereum/eth-vaults/convex-frax-asset/ERC1155Receipt.sol       | 69.23% (18/26)    | 60.61% (20/33)    | 41.67% (5/12)     | 50.00% (6/12)    |
| test/mocks/GeneralMatureHoldingVault.sol                           | 80.00% (12/15)    | 80.00% (12/15)    | 75.00% (3/4)      | 66.67% (8/12)    |
| Total                                                              | 50.94% (216/424)  | 46.72% (235/503)  | 50.00% (65/130)   | 53.10% (60/113)  |
```

Fuzzing

At the commit hash 59ff2b0c5841843ea6cae09927bd24d5fbb8e2a3, Arcadia wrote Foundry-based fuzzing tests. This is a harmless deviation from the normal methodology of using Echidna in order to keep compatibility between the existing usage of Foundry elsewhere in the codebase.

```
Running 6 tests for
test/RewardsDistributorFuzzTest.t.sol:RewardsDistributorFuzzTest
[PASS] testFuzz_DefaultAdminCanSetNewRoles(address) (runs: 256, μ: 49199,
~: 49199)
[PASS]
testFuzz_NonCheckpointerCannotCheckpointClaimPeriod(address,uint16,uint8,ui
nt256[8],bytes32[8]) (runs: 256, μ: 1506145, ~: 1647811)
[PASS] testFuzz_NonSystemCannotRegister1155(address,address) (runs: 256, μ:
42681, ~: 42681)
[PASS] testFuzz_NonUpgraderCannotFeeManager(address,address) (runs: 256, μ:
42560, ~: 42560)
[PASS] testFuzz_SystemCanRegister1155(address,address) (runs: 256, μ:
75425, ~: 75425)
[PASS] testFuzz_UpgraderCanSetFeeManager(address,address) (runs: 256, μ:
56819, ~: 56838)
Test result: ok. 6 passed; 0 failed; finished in 1119.54s

Running 14 tests for test/FeeManagerFuzzTest.t.sol:FeeManagerFuzzTest
[PASS] testFuzz_CannotCallInitOnProxyOrImpl(address) (runs: 256, μ: 25526,
~: 25526)
[PASS]
testFuzz_CannotSetAbove50PctFees(address,uint16,uint16,uint16,uint16)
(runs: 256, μ: 48126, ~: 48126)
[PASS] testFuzz_DefaultAdminCanSetNewRoles(address) (runs: 256, μ: 49499,
~: 49499)
[PASS] testFuzz_NonSetterCannotFeeAddress(address,address) (runs: 256, μ:
66378, ~: 66378)
[PASS] testFuzz_NonSetterCannotFees(address,uint16,uint16,uint16,uint16)
(runs: 256, μ: 66304, ~: 66304)
[PASS] testFuzz_NonSetterCannotRewardsAddress(address,address,address)
(runs: 256, μ: 66458, ~: 66458)
[PASS] testFuzz_NonUpgraderCannotUpgradeImpl(address) (runs: 256, μ:
2004743, ~: 2004743)
[PASS] testFuzz_RoleHolderCannotSetNewRoles(address,address) (runs: 256, μ:
65515, ~: 65515)
[PASS]
testFuzz_SetRewardsAddressIgnoreZeroRewardDistributor(address,address,addre
ss) (runs: 256, μ: 67026, ~: 67026)
[PASS]
testFuzz_SetRewardsAddressIgnoreZeroRewardsAddress(address,address,address)
(runs: 256, μ: 66922, ~: 66922)
[PASS] testFuzz_SetterCanSetFeeAddress(address,address) (runs: 256, μ:
54622, ~: 54641)
[PASS] testFuzz_SetterCanSetFees(address,uint16,uint16,uint16,uint16)
(runs: 256, μ: 104536, ~: 111159)
[PASS] testFuzz_SetterCanSetRewardsAddress(address,address,address) (runs:
256, μ: 93880, ~: 93880)
[PASS] testFuzz_UpgraderCanUpgradeImpl(address) (runs: 256, μ: 1994017, ~:
1994017)
Test result: ok. 14 passed; 0 failed; finished in 1256.16s

Running 3 tests for
test/HourglassCustodianFuzzTest.t.sol:HourglassCustodianFuzzTest
[PASS] testFuzz_AddPauserToCustodian(address) (runs: 256, μ: 56797, ~:
56783)
[PASS] testFuzz_CanRescueTokenFromVault(uint256,uint256) (runs: 256, μ:
854937, ~: 857036)
[PASS]
testFuzz_CannotRescueTokenFromVaultAsNonCustodian(address,uint256,uint256)
(runs: 256, μ: 823920, ~: 823920)
Test result: ok. 3 passed; 0 failed; finished in 1360.93s

Running 41 tests for test/HourglassCustodianTest.t.sol:CustodianTest
[PASS] testAddPauserToCustodian() (gas: 58677)
[PASS] testCanCallGetters() (gas: 139235)
```

```
[PASS] testCanPauseAsPauser() (gas: 41257)
[PASS] testCanPauseDistroClaiming() (gas: 146263)
[PASS] testCanRescueTokenFromVault() (gas: 249434)
[PASS] testCanSetDistroFeeManagerAsSetter() (gas: 22353)
[PASS] testCanSetFeeManagerFeesAsSetter() (gas: 70184)
[PASS] testCanTransferAndRedeemReceipts() (gas: 6943116)
[PASS] testCanTransferIntermediaryOwner() (gas: 29284)
[PASS] testCanTransferMatureVaultOwner() (gas: 31964)
[PASS] testCannotAddAssetAsNonAdmin() (gas: 66089)
[PASS] testCannotCallInitOnProxyOrImpl() (gas: 28718)
[PASS] testCannotCheckpointReceiptAsNonReceipt() (gas: 53049)
[PASS] testCannotDeployExistingVault() (gas: 49182)
[PASS] testCannotDeployTrancheAsNonVaultDeployer() (gas: 5306539)
[PASS] testCannotDepositBeyondMaxDuration() (gas: 8650211)
[PASS] testCannotPauseAsNonPauser() (gas: 56788)
[PASS] testCannotRedeemMoreThanCurrentBalance() (gas: 1074885)
[PASS] testCannotRegister1155AsNonSystem() (gas: 41190)
[PASS] testCannotReinitFeeManager() (gas: 19982)
[PASS] testCannotReinitRewardDistro() (gas: 20082)
[PASS] testCannotRescueTokenFromVaultAsNonCustodian() (gas: 115781)
[PASS] testCannotResuceTokensAsNonRescuer() (gas: 148117)
[PASS] testCannotSetDistroFeeManagerAsNonSetter() (gas: 41753)
[PASS] testCannotSetFeeAddressAsNonSetter() (gas: 42556)
[PASS] testCannotSetFeeManagerFeesAsNonSetter() (gas: 41630)
[PASS] testCannotSetRewardsAddressAsNonSetter() (gas: 42590)
[PASS] testCannotTransferIntermediaryOwnerAsNonOwner() (gas: 56421)
[PASS] testCannotTransferMatureVaultOwnerAsNonOwner() (gas: 66112)
[PASS] testCannotUnpaseAsNonPauser() (gas: 77454)
[PASS] testCannotUpgradeMatureVaultToZeroAddress() (gas: 17563)
[PASS] testDefaultAdminCanSetNewRoles() (gas: 50232)
[PASS] testDepositByCreatingVaultAndClaimingRewards() (gas: 3180290)
[PASS] testDepositWithPredeployedVaults() (gas: 1015498)
[PASS] testReleaseLocks() (gas: 11886529)
[PASS] testRescueTokens() (gas: 178647)
[PASS] testRoleHolderCannotSetNewRoles() (gas: 73059)
[PASS] testSetFeeAddressAsSetter() (gas: 22374)
[PASS] testSetRewardsAddressAsSetter() (gas: 25239)
[PASS] testUnpauseAsPauser() (gas: 32604)
[PASS] testUpgradeMatureVault() (gas: 775681)
Test result: ok. 41 passed; 0 failed; finished in 1376.43s
```

Conclusion

Arcadia identified issues that occurred at different commit hashes til #074648cdf715000e32685f00faa6fce4cd690856. This has been done a few times iteratively due to the changes frequently made to the contracts during the audit.

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.