Aura Finance

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **May 16th, 2022 - June 28th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	06/20/2022	István Böhm
0.2	Document Updates	06/28/2022	István Böhm
0.3	Draft Review	06/30/2022	Gabi Urrutia
1.0	Remediation Plan	07/01/2022	István Böhm
1.1	Remediation Plan Review	07/01/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
István Böhm	Halborn	Istvan.Bohm@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Aura Finance engaged Halborn to conduct a security audit on their smart contracts beginning on May 16th, 2022 and ending on June 28th, 2022. The security assessment was scoped to the smart contracts provided in the contracts GitHub repository aurafinance/aura-contracts.

1.2 AUDIT SUMMARY

The team at Halborn was provided six weeks for the engagement and assigned one full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified few security risks that were accepted and acknowledged by the Aura Finance team.

1.3 TEST APPROACH & 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 audit. 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 audit:

- 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
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment (Brownie, Remix IDE)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.

3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to the following smart contracts:

aura-contracts:
- Aura.sol
- AuraBalRewardPool.sol
- AuraClaimZap.sol
- AuraLocker.sol
- AuraMath.sol
- AuraMerkleDrop.sol
- AuraMinter.sol
- AuraPenaltyForwarder.sol
- AuraStakingProxy.sol
- AuraVestedEscrow.sol
- BalInvestor.sol
- BalLiquidityProvider.sol
- CrvDepositorWrapper.sol
- ExtraRewardsDistributor.sol
- RewardPoolDepositWrapper.sol

convex-platform:
- BaseRewardPool.sol
- VirtualBalanceRewardPool.sol
- ProxyFactory.sol
- DepositToken.sol
- ExtraRewardStashV3.sol
- RewardFactory.sol
- cCrv.sol
- BaseRewardPool4626.sol
- StashFactoryV2.sol
- PoolManagerSecondaryProxy.sol
- VoterProxy.sol
- Interfaces.sol
- TokenFactory.sol
- PoolManagerProxy.sol

EXECUTIVE OVERVIEW

- CrvDepositor.sol
- Booster.sol
- ConvexMasterChef.sol
- BoosterOwner.sol
- RewardHook.sol
- PoolManagerV3.sol
- ArbitartorVault.sol

Commit ID: b67d5b7d7fb87455533b5376e7c20157a6fc4e8c

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	4

LIKELIHOOD

IMPACT

(HAL-01)				
(HAL-03) (HAL-04) (HAL-05) (HAL-06)		(HAL-02)		

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - LACK OF TRANSFEROWNERSHIP PATTERN	Low	RISK ACCEPTED
HAL02 - DUPLICATE ENTRY IN THE VESTING DISTRIBUTION LIST	Low	RISK ACCEPTED
HAL03 - MISTAKENLY SENT ERC20 TOKENS CAN NOT RESCUED IN THE CONTRACTS	Informational	ACKNOWLEDGED
HAL04 - USING POSTFIX OPERATORS IN LOOPS	Informational	ACKNOWLEDGED
HAL05 - ARRAY.LENGTH USED IN LOOP CONDITIONS	Informational	ACKNOWLEDGED
HAL06 - USING != 0 CONSUMES LESS GAS THAN > 0 IN UNSIGNED INTEGER VALIDATION	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) LACK OF TRANSFEROWNERSHIP PATTERN - LOW

Description:

The current ownership transfer process for the Aura contracts inheriting from Ownable involves the current owner calling the transferOwnership() function:

Listing 1: Ownable.sol

```
71     function _transferOwnership(address newOwner) internal virtual
↳  {
72         address oldOwner = _owner;
73         _owner = newOwner;
74         emit OwnershipTransferred(oldOwner, newOwner);
75     }
76 }
```

If the nominated account is not a valid account, it is entirely possible that the owner may accidentally transfer ownership to an uncontrolled account, losing the access to all functions with the onlyOwner modifier. For example, in the case of the AuraLocker contract, if a not valid account was assigned as a owner, administrative functions such as recovering LP rewards from other systems or shutting down the contract will not be possible.

This issue also applies to other types of privilege transfer methods, like the setAdmin function in the AuraVestedEscrow contract:

Listing 2: AuraVestedEscrow.sol (Line 206)

```
79     function setAdmin(address _admin) external {
80         require(msg.sender == admin, "!auth");
81         admin = _admin;
82     }
```

Affected Contracts:
- aura-contracts/AuraClaimZap.sol
- aura-contracts/AuraLocker.sol
- aura-contracts/AuraPenaltyForwarder.sol
- aura-contracts/ExtraRewardsDistributor.sol
- aura-contracts/AuraVestedEscrow.sol
- convex-platform/Booster.sol
- convex-platform/ConvexMasterChef.sol

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to implement a two-step process where the owner nominates an account and the nominated account needs to call an acceptOwnership() function for the transfer of the ownership to fully succeed. This ensures the nominated account is a valid and active account.

Remediation Plan:

RISK ACCEPTED: The Aura Finance team accepted the risk of this finding and does not plan to correct it in the future in order to keep the difference between Aura and Convex as minimal as possible to aid in manual reviews and minimize the chance of introducing bugs.

3.2 (HAL-02) DUPLICATE ENTRY IN THE VESTING DISTRIBUTION LIST - LOW

Description:

0xcc6548f1b572968f9539d604ec9ff4b933c1be74 address accidentally appeared twice in the AURA vesting distribution list (tasks/deploy/mainnet-config.ts).

Listing 3: tasks/deploy/mainnet-config.ts

```
170 // 24 MONTHS - 8.45%
171 {
172     period: ONE_WEEK.mul(104),
173     recipients: [
174         { address: "0xe3B6c287C1369C6A4fa8d4e857813695C52948EF",
 ↳ amount: simpleToExactAmount(0.275, 24) }, // Core team
175         { address: "0x023320e0C9Ac45644c3305cE574360E901c7f582",
 ↳ amount: simpleToExactAmount(0.5, 24) }, // Core team
176         { address: "0xB1f881f47baB744E7283851bC090bAA626df931d",
 ↳ amount: simpleToExactAmount(3.5, 24) }, // Core team
177         { address: "0xE4b32828B558F17BcaF5efD52f0C067dba38833c",
 ↳ amount: simpleToExactAmount(0.45, 24) }, // Core team
178         { address: "0xcc6548f1b572968f9539d604ec9ff4b933c1be74",
 ↳ amount: simpleToExactAmount(0.075, 24) }, // Core team
179         { address: "0x51d63958a63a31eb4028917f049ce477c8dd07bb",
 ↳ amount: simpleToExactAmount(0.5, 24) }, // Core team
180         { address: "0x3078c3b436511152d86675f9cbfd89ec1672f804",
 ↳ amount: simpleToExactAmount(0.3, 24) }, // Core team
181         { address: "0x3000d9b2c0e6b9f97f30abe379eaaa8a85a04afc",
 ↳ amount: simpleToExactAmount(0.325, 24) }, // Core team
182         { address: "0x3CBFFF3E75881c1619eaa82DC724BDEE6fF6ED19",
 ↳ amount: simpleToExactAmount(0.06, 24) }, // Core team
183         { address: "0xaf3824e8401299B25C4D59a8a035Cf9312a3B454",
 ↳ amount: simpleToExactAmount(0.175, 24) }, // Core team
184         { address: "0x738175DB2C999581f29163e6D4D3516Ad4aF8834",
 ↳ amount: simpleToExactAmount(0.125, 24) }, // Core team
185         { address: "0x0d9A5678E73e5BbC0ee09FAF8e550B196c76fDad",
 ↳ amount: simpleToExactAmount(0.5, 24) }, // Core team
186         { address: "0x285b7EEa81a5B66B62e7276a24c1e0F83F7409c1",
 ↳ amount: simpleToExactAmount(1.5, 24) }, // Core team
```

```
187         { address: "0xbee5a45271cc66a5b0e9dc4164a4f9df196d94fa",
 ↳ amount: simpleToExactAmount(0.125, 24) }, // Core team
188         { address: "0xcc6548f1b572968f9539d604ec9ff4b933c1be74",
 ↳ amount: simpleToExactAmount(0.04, 24) }, // Core team
189     ],
190 },
```

Listing 4: scripts/deploySystem.ts

```
700         const vestingAddr = vestingGroup.recipients.map(m => m.
 ↳ address);
701         const vestingAmounts = vestingGroup.recipients.map(m => m.
 ↳ amount);
702         tx = await vestedEscrow.fund(vestingAddr, vestingAmounts);
```

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

It is recommended reviewing the configuration data used to fund the vesting recipients and, if necessary, correcting the values using the vesting admin.

Remediation Plan:

RISK ACCEPTED: The Aura Finance team will correct this finding through the governance.

3.3 (HAL-03) MISTAKENLY SENT ERC20 TOKENS CAN NOT RESCUED IN THE CONTRACTS - INFORMATIONAL

Description:

The contracts are missing functions to sweep/rescue accidental ERC-20 transfers. Accidentally, sent ERC-20 tokens will be locked in the contracts.

Recommendation:

Consider adding a function to sweep accidental ERC-20 transfers to the contracts.

Remediation Plan:

ACKNOWLEDGED: The Aura Finance team acknowledged this finding and does not plan to fix it in the future to keep the difference between Aura and Convex as minimal as possible to aid in the manual reviews and minimize the chance of introducing bugs.

3.4 (HAL-04) USING POSTFIX OPERATORS IN LOOPS - INFORMATIONAL

Description:

In the loops below, postfix (e.g. i++) operators were used to increment or decrement variable values. It is known that, in loops, using prefix operators (e.g. ++i) costs less gas per iteration than using postfix operators.

Code Location:

aura-contracts/AuraClaimZap.sol
- Line 134 `for (uint256 i = 0; i < rewardContracts.length; i++){`
- Line 138 `for (uint256 i = 0; i < extraRewardContracts.length; i++){`
- Line 142 `for (uint256 i = 0; i < tokenRewardContracts.length; i++){`

aura-contracts/AuraLocker.sol
- Line 176 `for (uint256 i = 0; i < rewardTokensLength; i++){`
- Line 332 `for (uint256 i; i < rewardTokensLength; i++){`
- Line 350 `for (uint256 i; i < rewardTokensLength; i++){`
- Line 450 `for (uint256 i = nextUnlockIndex; i < length; i++){`
- Line 466 `nextUnlockIndex++;`
- Line 537 `i--;;`

aura-contracts/AuraVestedEscrow.sol
- Line 105 `for (uint256 i = 0; i < _recipient.length; i++){`

aura-contracts/BalLiquidityProvider.sol
- Line 52 `for (uint256 i = 0; i < 2; i++){`

aura-contracts/ExtraRewardsDistributor.sol
- Line 242 `for (uint256 i = epochIndex; i < tokenEpochs; i++){`

convex-platform/ArbitartorVault.sol
- Line 49 `for(uint256 i = 0; i < _toPids.length; i++){`

convex-platform/BaseRewardPool.sol
- Line 218 `for(uint i=0; i < extraRewards.length; i++){`
- Line 234 `for(uint i=0; i < extraRewards.length; i++){`
- Line 266 `for(uint i=0; i < extraRewards.length; i++){`
- Line 300 `for(uint i=0; i < extraRewards.length; i++){`

convex-platform/Booster.sol
- Line 380 `for(uint i=0; i < poolInfo.length; i++){`
- Line 539 `for(uint256 i = 0; i < _gauge.length; i++){`

convex-platform/BoosterOwner.sol
- Line 144 `for(uint256 i = 0; i < poolCount; i++){`

convex-platform/ExtraRewardStashV3.sol
- Line 125 `for(uint256 i = 0; i < maxRewards; i++){`
- Line 201 `for(uint i=0; i < tCount; i++){`

convex-platform/PoolManagerSecondaryProxy.sol
- Line 69 `for(uint i=0; i < usedList.length; i++){`

It is also possible to further optimize loops by using unchecked loop index incrementing and decrementing.

Proof of Concept:

For example, based on the following test contract:

Listing 5: GasTestIncrement.sol

```
1  //SPDX-License-Identifier: MIT
2  pragma solidity 0.8.11;
3
4  contract GasTestIncrement {
5      function postiincrement(uint256 iterations) public {
6          for (uint256 i = 0; i < iterations; i++) {
7          }
8      }
9      function preiincrement(uint256 iterations) public {
10         for (uint256 i = 0; i < iterations; ++i) {
11         }
12     }
13     function uncheckedpreiincrement(uint256 iterations) public {
14         for (uint256 i = 0; i < iterations;) {
15             unchecked { ++i; }
16         }
17     }
18 }
```

We can see the difference in gas costs:

```
>>> contract_gastest.postiincrement(10)
Transaction sent: 0xea37b9e304229d9063189c85f0a10f6d3aee232cb0e527d09ea9cc33ae5d29d9
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 8
  GasTestIncrement.postiincrement confirmed   Block: 14915970   Gas used: 22651 (0.34%)

<Transaction '0xea37b9e304229d9063189c85f0a10f6d3aee232cb0e527d09ea9cc33ae5d29d9'>
>>> contract_gastest.preiincrement(10)
Transaction sent: 0x1e688f5c8c7d3e393c52eb214f2278f7f561e55857b36c4b5c1d29ace0e6ce5d
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 9
  GasTestIncrement.preiincrement confirmed   Block: 14915971   Gas used: 22557 (0.34%)

<Transaction '0x1e688f5c8c7d3e393c52eb214f2278f7f561e55857b36c4b5c1d29ace0e6ce5d'>
>>> contract_gastest.uncheckedpreiincrement(10)
Transaction sent: 0xec50e014f2de0b3badd8269daaf26b90eef2315de8e8ba65799def941d058772
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 10
  GasTestIncrement.uncheckedpreiincrement confirmed   Block: 14915972   Gas used: 21889 (0.33%)

<Transaction '0xec50e014f2de0b3badd8269daaf26b90eef2315de8e8ba65799def941d058772'>
>>>
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use unchecked `++i` and `--j` operations instead of `i++` and `j--` to increment or decrement the values of a `uint` variables inside loops. This does not just apply to the iterator variables, but the increments and decrements done inside the loops code blocks too.

It is noted that using unchecked operations requires particular caution to avoid overflows.

Remediation Plan:

ACKNOWLEDGED: The Aura Finance team acknowledged this finding and does not plan to correct it in the future in order to keep the difference between Aura and Convex as minimal as possible to aid in manual reviews and minimize the chance of introducing bugs.

3.5 (HAL-05) ARRAY.LENGTH USED IN LOOP CONDITIONS - INFORMATIONAL

Description:

In the loops below, unnecessary reading of the lengths of arrays on each iteration wastes gas.

Code Location:

aura-contracts/AuraClaimZap.sol
- Line 134 `for (uint256 i = 0; i < rewardContracts.length; i++){`
- Line 138 `for (uint256 i = 0; i < extraRewardContracts.length; i++){`
- Line 142 `for (uint256 i = 0; i < tokenRewardContracts.length; i++){`

aura-contracts/AuraVestedEscrow.sol
- Line 105 `for (uint256 i = 0; i < _recipient.length; i++){`

convex-platform/ArbitartorVault.sol
- Line 49 `for(uint256 i = 0; i < _toPids.length; i++){`

convex-platform/BaseRewardPool.sol
- Line 218 `for(uint i=0; i < extraRewards.length; i++){`
- Line 234 `for(uint i=0; i < extraRewards.length; i++){`
- Line 266 `for(uint i=0; i < extraRewards.length; i++){`
- Line 300 `for(uint i=0; i < extraRewards.length; i++){`

convex-platform/Booster.sol
- Line 380 `for(uint i=0; i < poolInfo.length; i++){`
- Line 539 `for(uint256 i = 0; i < _gauge.length; i++){`

convex-platform/PoolManagerSecondaryProxy.sol
- Line 69 `for(uint i=0; i < usedList.length; i++){`

Proof of Concept:

For example, based on the following test contract:

Listing 6: GasTestLength.sol

```
1  //SPDX-License-Identifier: MIT
2  pragma solidity 0.8.11;
3
4  contract GasTestLength {
5
6      uint256[] private arr = [0,1,2,3,4,5,6,7,8,9];
7
8      function unoptimalized() public {
9          for (uint256 i = 0; i < arr.length; ++i) {
10         }
11
12     }
13     function optimalized() public {
14         uint256 length = arr.length;
15         for (uint256 i = 0; i < length; ++i) {
16         }
17     }
18 }
```

We can see the difference in gas costs:

```
>>> contract_gastest.unoptimalized()
Transaction sent: 0x5aa7c5bed5bf82bd8fc3af9daf8d8e9b140659bcab17aa3443cee4852f9484bc
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 1
  GasTestLength.unoptimalized confirmed   Block: 14915963   Gas used: 31138 (0.46%)

<Transaction '0x5aa7c5bed5bf82bd8fc3af9daf8d8e9b140659bcab17aa3443cee4852f9484bc'>
>>> contract_gastest.optimalized()
Transaction sent: 0x39409c65ce53a6c8f3a16a301fed90b146634095aaaf7a79a268ad4ea8f298e3
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 2
  GasTestLength.optimalized confirmed   Block: 14915964   Gas used: 23168 (0.34%)

<Transaction '0x39409c65ce53a6c8f3a16a301fed90b146634095aaaf7a79a268ad4ea8f298e3'>
>>>
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to cache array lengths outside of loops as long the size is not changed during the loop:

Listing 7

```
uint256 length = arr.length;
for (uint256 i = 0; i < length; ++i) {
    ...
}
```

Remediation Plan:

ACKNOWLEDGED: The Aura Finance team acknowledged this finding and does not plan to correct it in the future in order to keep the difference between Aura and Convex as minimal as possible to aid in manual reviews and minimize the chance of introducing bugs.

3.6 (HAL-06) USING != 0 CONSUMES LESS GAS THAN > 0 IN UNSIGNED INTEGER VALIDATION - INFORMATIONAL

Description:

In the require statements below, > 0 was used to validate if the unsigned integer parameters are bigger than 0. It is known that, using != 0 costs less gas than > 0.

Code Location:

aura-contracts/AuraBalRewardPool.sol
- Line 121 require(_amount > 0, "RewardPool : Cannot stake 0");
- Line 139 require(_amount > 0, "RewardPool : Cannot stake 0");
- Line 157 require(amount > 0, "RewardPool : Cannot withdraw 0");
- Line 232 require(rewardsAvailable > 0, "!balance");

aura-contracts/AuraLocker.sol
- Line 236 require(rewardData[_rewardsToken].lastUpdateTime > 0, ...
- Line 285 require(_amount > 0, "Cannot stake 0");
- Line 399 require(amt > 0, "Nothing locked");
- Line 425 require(length > 0, "no locks");
- Line 471 require(locked > 0, "no exp locks");
- Line 511 require(len > 0, "Nothing to delegate");
- Line 862 require(_rewards > 0, "No reward");

aura-contracts/AuraMerkleDrop.sol
- Line 139 require(_amount > 0, "!amount");

aura-contracts/AuraPenaltyForwarder.sol
- Line 55 require(bal > 0, "!empty");

aura-contracts/AuraVestedEscrow.sol
- Line 55 require(totalLocked[_recipient] > 0, "!funding");

aura-contracts/BalLiquidityProvider.sol
- Line 74 require(balAfter > 0, "!mint");

aura-contracts/ExtraRewardsDistributor.sol
- Line 104 require(_amount > 0, "!amount");
- Line 180 require(_index > 0 && ...);

aura-contracts/RewardPoolDepositWrapper.sol
- Line 51 ‘require(minted > 0, “!mint”);‘‘

convex-platform/BaseRewardPool.sol
- Line 215 require(_amount > 0, 'RewardPool : Cannot stake 0');
- Line 231 require(amount > 0, 'RewardPool : Cannot withdraw 0');

convex-platform/ConvexMasterChef.sol
- Line 138 require(totalAllocPoint > 0, "!alloc");

convex-platform/CrvDepositor.sol
- Line 169 require(_amount > 0,"!>0");

convex-platform/PoolManagerSecondaryProxy.sol
- Line 104 require(weight > 0, "must have weight");

convex-platform/interfaces/BoringMath.sol
- Line 20 require(b > 0, "BoringMath: division by zero");
- Line 102 require(b > 0, "BoringMath: division by zero");
- Line 123 require(b > 0, "BoringMath: division by zero");
- Line 143 require(b > 0, "BoringMath: division by zero");

Proof of Concept:

For example, based on the following test contract:

Listing 8: GasTestRequire.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity 0.8.11;
3
4 contract GasTestRequire {
5     function originalrequire(uint256 len) public {
6         require(len > 0, "Error!");
7     }
8     function optimalizedrequire(uint256 len) public {
9         require(len != 0, "Error!");
10    }
11 }
```

We can see the difference in gas costs:

```
>>> contract_gastest.originalrequire(10)
Transaction sent: 0x3bfd50e87f0b7baa6d546f38a78b9a0332cc45b8639ab7e6a641878d46df5b3c
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 5
  GasTestRequire.originalrequire confirmed   Block: 14915967   Gas used: 21450 (0.32%)

<Transaction '0x3bfd50e87f0b7baa6d546f38a78b9a0332cc45b8639ab7e6a641878d46df5b3c'>
>>> contract_gastest.optimalizedrequire(10)
Transaction sent: 0x7ce50b4e528cd2b0488254a3bbd6ca08e04d7ac29753f8513d6d169ee5e190e3
  Gas price: 0.0 gwei   Gas limit: 6721975   Nonce: 6
  GasTestRequire.optimalizedrequire confirmed   Block: 14915968   Gas used: 21422 (0.32%)

<Transaction '0x7ce50b4e528cd2b0488254a3bbd6ca08e04d7ac29753f8513d6d169ee5e190e3'>
>>>
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to use `!= 0` instead of `> 0` to validate unsigned integer parameters. For example, use instead:

Listing 9

```
1 require(_amount != 0, "RewardPool : Cannot stake 0");
```

Remediation Plan:

ACKNOWLEDGED: The Aura Finance team acknowledged this finding and does not plan to correct it in the future to keep the difference between Aura and Convex as minimal as possible to aid in manual reviews and minimize the chance of introducing bugs.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on the all-scoped 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.

Slither results:

Note that due to the significant number of contracts, the low-risk findings displayed by Slither were not included in the report. However, we examined them individually during our audit.

aura-contracts/AuraLocker.sol

```
AuraLocker.constructor(string,string,address,address,address) (contracts/AuraLocker.sol#149-166) performs a multiplication on the result of a division:
        -currentEpoch = block.timestamp.div(rewardsDuration).mul(rewardsDuration) (contracts/AuraLocker.sol#164)
AuraLocker._lock(address,uint256) (contracts/AuraLocker.sol#284-322) performs a multiplication on the result of a division:
        -currentEpoch = block.timestamp.div(rewardsDuration).mul(rewardsDuration) (contracts/AuraLocker.sol#301)
AuraLocker._checkpointEpoch() (contracts/AuraLocker.sol#367-379) performs a multiplication on the result of a division:
        -currentEpoch = block.timestamp.div(rewardsDuration).mul(rewardsDuration) (contracts/AuraLocker.sol#368)
AuraLocker._processExpiredLocks(address,bool,address,uint256) (contracts/AuraLocker.sol#411-498) performs a multiplication on the result of a division:
        -currentEpoch = block.timestamp.sub(_checkDelay).div(rewardsDuration).mul(rewardsDuration) (contracts/AuraLocker.sol#441)
AuraLocker._processExpiredLocks(address,bool,address,uint256) (contracts/AuraLocker.sol#411-498) performs a multiplication on the result of a division:
        -currentEpoch_scope_0 = block.timestamp.sub(_checkDelay).div(rewardsDuration).mul(rewardsDuration) (contracts/AuraLocker.sol#460)
AuraLocker.delegate(address) (contracts/AuraLocker.sol#507-549) performs a multiplication on the result of a division:
        -upcomingEpoch = block.timestamp.add(rewardsDuration).div(rewardsDuration).mul(rewardsDuration) (contracts/AuraLocker.sol#523)
AuraLocker._checkpointDelegate(address,uint256,uint256) (contracts/AuraLocker.sol#551-604) performs a multiplication on the result of a division:
        -upcomingEpoch = block.timestamp.add(rewardsDuration).div(rewardsDuration).mul(rewardsDuration) (contracts/AuraLocker.sol#558)
AuraLocker.getPastVotes(address,uint256) (contracts/AuraLocker.sol#637-649) performs a multiplication on the result of a division:
        -epoch = timestamp.div(rewardsDuration).mul(rewardsDuration) (contracts/AuraLocker.sol#639)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

AuraLocker._checkpointDelegate(address,uint256,uint256) (contracts/AuraLocker.sol#551-604) uses a dangerous strict equality:
        - prevCkpt.epochStart == upcomingEpoch (contracts/AuraLocker.sol#563)
AuraLocker.totalSupplyAtEpoch(uint256) (contracts/AuraLocker.sol#757-776) uses a dangerous strict equality:
        - e.date == epochStart (contracts/AuraLocker.sol#768)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in AuraLocker._lock(address,uint256) (contracts/AuraLocker.sol#284-322):
        External calls:
        - _checkpointDelegate(delegatee,lockAmount,0) (contracts/AuraLocker.sol#314)
                - ckpts[ckpts.length - 1] = DelegateeCheckpoint((prevCkpt.votes + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32()) (contracts/AuraLocker.sol#564-567)
                - ckpts.push(DelegateeCheckpoint((prevCkpt.votes - unlocksSinceLatestCkpt + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#586-592)
        State variables written after the call(s):
        - e.supply = e.supply.add(lockAmount) (contracts/AuraLocker.sol#319)
Reentrancy in AuraLocker._processExpiredLocks(address,bool,address,uint256) (contracts/AuraLocker.sol#411-498):
        External calls:
        - _checkpointDelegate(delegates(_account),0,0) (contracts/AuraLocker.sol#478)
                - ckpts[ckpts.length - 1] = DelegateeCheckpoint((prevCkpt.votes + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32()) (contracts/AuraLocker.sol#564-567)
                - ckpts.push(DelegateeCheckpoint((prevCkpt.votes - unlocksSinceLatestCkpt + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#586-592)
        - stakingToken.safeTransfer(_rewardAddress,reward) (contracts/AuraLocker.sol#488)
        - _lock(_account,locked) (contracts/AuraLocker.sol#494)
                - ckpts[ckpts.length - 1] = DelegateeCheckpoint((prevCkpt.votes + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32()) (contracts/AuraLocker.sol#564-567)
                - ckpts.push(DelegateeCheckpoint((prevCkpt.votes - unlocksSinceLatestCkpt + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#586-592)
        State variables written after the call(s):
        - _lock(_account,locked) (contracts/AuraLocker.sol#494)
                - ckpts[ckpts.length - 1] = DelegateeCheckpoint((prevCkpt.votes + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32()) (contracts/AuraLocker.sol#564-567)
                - ckpts.push(DelegateeCheckpoint((_upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#572-577)
                - ckpts.push(DelegateeCheckpoint((prevCkpt.votes - unlocksSinceLatestCkpt + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#586-592)
                - ckpts.push(DelegateeCheckpoint((_upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#595-600)
        - _lock(_account,locked) (contracts/AuraLocker.sol#494)
                - bal.locked = bal.locked.add(lockAmount) (contracts/AuraLocker.sol#295)
        - _lock(_account,locked) (contracts/AuraLocker.sol#494)
                - delegateeUnlocks[delegatee][unlockTime] += lockAmount (contracts/AuraLocker.sol#313)
        - _lock(_account,locked) (contracts/AuraLocker.sol#494)
                - lockedSupply = lockedSupply.add(_amount) (contracts/AuraLocker.sol#298)
        - _lock(_account,locked) (contracts/AuraLocker.sol#494)
                - userLocks[_account].push(LockedBalance(lockAmount,uint32(unlockTime))) (contracts/AuraLocker.sol#305)
                - userL.amount = userL.amount.add(lockAmount) (contracts/AuraLocker.sol#308)
```

```
Reentrancy in AuraLocker.delegate(address) (contracts/AuraLocker.sol#507-549):
        External calls:
        - _checkpointDelegate(oldDelegatee,0,futureUnlocksSum) (contracts/AuraLocker.sol#545)
                - ckpts[ckpts.length - 1] = DelegateeCheckpoint((prevCkpt.votes + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32()) (contracts/AuraLocker.sol#564-567)
                - ckpts.push(DelegateeCheckpoint((prevCkpt.votes - unlocksSinceLatestCkpt + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#586-592)
        - _checkpointDelegate(newDelegatee,futureUnlocksSum,0) (contracts/AuraLocker.sol#548)
                - ckpts[ckpts.length - 1] = DelegateeCheckpoint((prevCkpt.votes + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32()) (contracts/AuraLocker.sol#564-567)
                - ckpts.push(DelegateeCheckpoint((prevCkpt.votes - unlocksSinceLatestCkpt + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#586-592)
        State variables written after the call(s):
        - _checkpointDelegate(newDelegatee,futureUnlocksSum,0) (contracts/AuraLocker.sol#548)
                - ckpts[ckpts.length - 1] = DelegateeCheckpoint((prevCkpt.votes + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32()) (contracts/AuraLocker.sol#564-567)
                - ckpts.push(DelegateeCheckpoint((_upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#572-577)
                - ckpts.push(DelegateeCheckpoint((prevCkpt.votes - unlocksSinceLatestCkpt + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#586-592)
                - ckpts.push(DelegateeCheckpoint((_upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#595-600)
Reentrancy in AuraLocker.lock(address,uint256) (contracts/AuraLocker.sol#275-281):
        External calls:
        - stakingToken.safeTransferFrom(msg.sender,address(this),_amount) (contracts/AuraLocker.sol#277)
        - _lock(_account,_amount) (contracts/AuraLocker.sol#280)
                - ckpts[ckpts.length - 1] = DelegateeCheckpoint((prevCkpt.votes + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32()) (contracts/AuraLocker.sol#564-567)
                - ckpts.push(DelegateeCheckpoint((prevCkpt.votes - unlocksSinceLatestCkpt + _upcomingAddition - _upcomingDeduction).to224(),upcomingEpoch.to32())) (contracts/AuraLocker.sol#586-592)
        State variables written after the call(s):
        - _lock(_account,_amount) (contracts/AuraLocker.sol#280)
                - bal.locked = bal.locked.add(lockAmount) (contracts/AuraLocker.sol#295)
        - _lock(_account,_amount) (contracts/AuraLocker.sol#280)
                - lockedSupply = lockedSupply.add(_amount) (contracts/AuraLocker.sol#298)
Reentrancy in AuraLocker.queueNewRewards(address,uint256) (contracts/AuraLocker.sol#860-888):
        External calls:
        - IERC20(_rewardsToken).safeTransferFrom(msg.sender,address(this),_rewards) (contracts/AuraLocker.sol#866)
        State variables written after the call(s):
        - _notifyReward(_rewardsToken,_rewards) (contracts/AuraLocker.sol#872)
                - rdata.rewardRate = _reward.div(rewardsDuration).to96() (contracts/AuraLocker.sol#894)
                - rdata.rewardRate = _reward.add(leftover).div(rewardsDuration).to96() (contracts/AuraLocker.sol#898)
                - rewardData[token].rewardPerTokenStored = newRewardPerToken.to96() (contracts/AuraLocker.sol#179)
                - rewardData[token].lastUpdateTime = _lastTimeRewardApplicable(rewardData[token].periodFinish).to32() (contracts/AuraLocker.sol#180)
        - _notifyReward(_rewardsToken,_rewards) (contracts/AuraLocker.sol#883)
                - rdata.rewardRate = _reward.div(rewardsDuration).to96() (contracts/AuraLocker.sol#894)
                - rdata.rewardRate = _reward.add(leftover).div(rewardsDuration).to96() (contracts/AuraLocker.sol#898)
                - rewardData[token].rewardPerTokenStored = newRewardPerToken.to96() (contracts/AuraLocker.sol#179)
                - rewardData[token].lastUpdateTime = _lastTimeRewardApplicable(rewardData[token].periodFinish).to32() (contracts/AuraLocker.sol#180)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

AuraLocker.lockedBalances(address).idx (contracts/AuraLocker.sol#735) is a local variable never initialized
AuraLocker.getReward(address,bool).i (contracts/AuraLocker.sol#332) is a local variable never initialized
AuraLocker.getReward(address,bool[]).i (contracts/AuraLocker.sol#350) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

aura-contracts/AuraVestedEscrow.sol

```
Reentrancy in AuraVestedEscrow.cancel(address) (contracts/AuraVestedEscrow.sol#121-133):
        External calls:
        - _claim(_recipient,false) (contracts/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/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/AuraVestedEscrow.sol#192)
                - rewardToken.safeTransfer(_recipient,claimable) (contracts/AuraVestedEscrow.sol#194)
        - rewardToken.safeTransfer(admin,delta) (contracts/AuraVestedEscrow.sol#128)
        External calls sending eth:
        - _claim(_recipient,false) (contracts/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/AuraVestedEscrow.sol#130)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

Reentrancy in AuraVestedEscrow.fund(address[],uint256[]) (contracts/AuraVestedEscrow.sol#98-115):
        External calls:
        - rewardToken.safeTransferFrom(msg.sender,address(this),totalAmount) (contracts/AuraVestedEscrow.sol#113)
        State variables written after the call(s):
        - initialised = true (contracts/AuraVestedEscrow.sol#114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

aura-contracts/BalInvestor.sol

```
BalInvestor._getMinOut(uint256,uint256) (contracts/BalInvestor.sol#58-66) performs a multiplication on the result of a division:
        -minOut = (((amount * 1e18) / bptOraclePrice) * minOutBps) / 10000 (contracts/BalInvestor.sol#64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

aura-contracts/CrvDepositorWrapper.sol

```
BalInvestor._getMinOut(uint256,uint256) (contracts/BalInvestor.sol#58-66) performs a multiplication on the result of a division:
        -minOut = (((amount * 1e18) / bptOraclePrice) * minOutBps) / 10000 (contracts/BalInvestor.sol#64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

aura-contracts/RewardPoolDepositWrapper.sol

```
RewardPoolDepositWrapper.depositSingle(address,IERC20,uint256,bytes32,IVault.JoinPoolRequest) (contracts/RewardPoolDepositWrapper.sol#34-59) uses a dangerous strict equality:
        - require(bool,string)(inputBalAfter == 0,Input) (contracts/RewardPoolDepositWrapper.sol#54)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

RewardPoolDepositWrapper.depositSingle(address,IERC20,uint256,bytes32,IVault.JoinPoolRequest) (contracts/RewardPoolDepositWrapper.sol#34-59) ignores return value by _inputToken.approve(address(bVault),_inputAmount) (contracts/RewardPoolDepositWrapper.sol#47)
RewardPoolDepositWrapper.depositSingle(address,IERC20,uint256,bytes32,IVault.JoinPoolRequest) (contracts/RewardPoolDepositWrapper.sol#34-59) ignores return value by IERC20(pool).approve(_rewardPoolAddress,minted) (contracts/RewardPoolDepositWrapper.sol#57)
RewardPoolDepositWrapper.depositSingle(address,IERC20,uint256,bytes32,IVault.JoinPoolRequest) (contracts/RewardPoolDepositWrapper.sol#34-59) ignores return value by RewardPool(_rewardPoolAddress).deposit(minted,msg.sender) (contracts/RewardPoolDepositWrapper.sol#58)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

convex-platform/BaseRewardPool.sol

```
BaseRewardPool.notifyRewardAmount(uint256) (contracts/convex-platform/BaseRewardPool.sol#366-383) performs a multiplication on the result of a division:
        -rewardRate = reward.div(duration) (contracts/convex-platform/BaseRewardPool.sol#372)
        -leftover = remaining.mul(rewardRate) (contracts/convex-platform/BaseRewardPool.sol#375)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

BaseRewardPool.rewardPerToken() (contracts/convex-platform/BaseRewardPool.sol#155-167) uses a dangerous strict equality:
        - totalSupply() == 0 (contracts/convex-platform/BaseRewardPool.sol#156)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
```

convex-platform/BaseRewardPool4626.sol

```
BaseRewardPool4626.deposit(uint256,address) (contracts/convex-platform/BaseRewardPool4626.sol#56-73) ignores return value by IDeposit(operator).deposit(pid,assets,false) (contracts/convex-platform/BaseRewardPool4626.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

convex-platform/Booster.sol

```
Reentrancy in Booster.addPool(address,address,uint256) (contracts/convex-platform/Booster.sol#313-351):
        External calls:
        - token = ITokenFactory(tokenFactory).CreateDepositToken(_lptoken) (contracts/convex-platform/Booster.sol#322)
        - newRewardPool = IRewardFactory(rewardFactory).CreateCrvRewards(pid,token,_lptoken) (contracts/convex-platform/Booster.sol#324)
        - stash = IStashFactory(stashFactory).CreateStash(pid,_gauge,staker,_stashVersion) (contracts/convex-platform/Booster.sol#326)
        State variables written after the call(s):
        - poolInfo.push(PoolInfo(_lptoken,token,_gauge,newRewardPool,stash,false)) (contracts/convex-platform/Booster.sol#329-338)
        - poolInfo[pid].stash = stash (contracts/convex-platform/Booster.sol#344)
Reentrancy in Booster.setFeeInfo(address,address) (contracts/convex-platform/Booster.sol#217-251):
        External calls:
        - rewards = IRewardFactory(rewardFactory).CreateTokenRewards(_feeToken,lockRewards,address(this)) (contracts/convex-platform/Booster.sol#239)
        State variables written after the call(s):
        - feeTokens[_feeToken] = FeeDistro(_feeDistro,rewards,true) (contracts/convex-platform/Booster.sol#240-244)
Reentrancy in Booster.shutdownPool(uint256) (contracts/convex-platform/Booster.sol#357-370):
        External calls:
        - IStaker(staker).withdrawAll(pool.lptoken,pool.gauge) (contracts/convex-platform/Booster.sol#362-363)
        State variables written after the call(s):
        - pool.shutdown = true (contracts/convex-platform/Booster.sol#365)
Reentrancy in Booster.shutdownSystem() (contracts/convex-platform/Booster.sol#376-392):
        External calls:
        - IStaker(staker).withdrawAll(token,gauge) (contracts/convex-platform/Booster.sol#388-390)
        State variables written after the call(s):
        - pool.shutdown = true (contracts/convex-platform/Booster.sol#389)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Booster.addPool(address,address,uint256) (contracts/convex-platform/Booster.sol#313-351) ignores return value by IStaker(staker).setStashAccess(stash,true) (contracts/convex-platform/Booster.sol#345)
Booster.shutdownPool(uint256) (contracts/convex-platform/Booster.sol#357-370) ignores return value by IStaker(staker).withdrawAll(pool.lptoken,pool.gauge) (contracts/convex-platform/Booster.sol#362-363)
Booster.shutdownSystem() (contracts/convex-platform/Booster.sol#376-392) ignores return value by IStaker(staker).withdrawAll(token,gauge) (contracts/convex-platform/Booster.sol#388-390)
Booster.deposit(uint256,uint256,bool) (contracts/convex-platform/Booster.sol#398-434) ignores return value by IStaker(staker).deposit(lptoken,gauge) (contracts/convex-platform/Booster.sol#410)
Booster.deposit(uint256,uint256,bool) (contracts/convex-platform/Booster.sol#398-434) ignores return value by IStash(stash).stashRewards() (contracts/convex-platform/Booster.sol#415)
Booster._withdraw(uint256,uint256,address,address) (contracts/convex-platform/Booster.sol#454-480) ignores return value by IStaker(staker).withdraw(lptoken,gauge,_amount) (contracts/convex-platform/Booster.sol#466)
Booster._withdraw(uint256,uint256,address,address) (contracts/convex-platform/Booster.sol#454-480) ignores return value by IStash(stash).stashRewards() (contracts/convex-platform/Booster.sol#473)
Booster.vote(uint256,address,bool) (contracts/convex-platform/Booster.sol#525-531) ignores return value by IStaker(staker).vote(_voteId,_votingAddress,_support) (contracts/convex-platform/Booster.sol#529)
Booster.voteGaugeWeight(address[],uint256[]) (contracts/convex-platform/Booster.sol#536-543) ignores return value by IStaker(staker).voteGaugeWeight(_gauge[i],_weight[i]) (contracts/convex-platform/Booster.sol#540)
Booster.claimRewards(uint256,address) (contracts/convex-platform/Booster.sol#548-554) ignores return value by IStaker(staker).claimRewards(_gauge) (contracts/convex-platform/Booster.sol#552)
Booster.setGaugeRedirect(uint256) (contracts/convex-platform/Booster.sol#559-566) ignores return value by IStaker(staker).execute(gauge,uint256(0),data) (contracts/convex-platform/Booster.sol#564)
Booster._earmarkRewards(uint256) (contracts/convex-platform/Booster.sol#573-628) ignores return value by IStaker(staker).claimCrv(gauge) (contracts/convex-platform/Booster.sol#580)
Booster._earmarkRewards(uint256) (contracts/convex-platform/Booster.sol#573-628) ignores return value by IStash(stash).claimRewards() (contracts/convex-platform/Booster.sol#586)
Booster._earmarkRewards(uint256) (contracts/convex-platform/Booster.sol#573-628) ignores return value by IStash(stash).processStash() (contracts/convex-platform/Booster.sol#588)
Booster.earmarkFees(address) (contracts/convex-platform/Booster.sol#645-663) ignores return value by IStaker(staker).claimFees(feeDistro.distro,_feeToken) (contracts/convex-platform/Booster.sol#654)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

convex-platform/ConvexMasterChef.sol

```
Reentrancy in ConvexMasterChef.claim(uint256,address) (contracts/convex-platform/ConvexMasterChef.sol#268-286):
        External calls:
        - safeRewardTransfer(_account,pending) (contracts/convex-platform/ConvexMasterChef.sol#276)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-0.6/token/ERC20/SafeERC20.sol#69)
                - cvx.safeTransfer(_to,cvxBal) (contracts/convex-platform/ConvexMasterChef.sol#308)
                - cvx.safeTransfer(_to,_amount) (contracts/convex-platform/ConvexMasterChef.sol#310)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-0.6/utils/Address.sol#119)
        External calls sending eth:
        - safeRewardTransfer(_account,pending) (contracts/convex-platform/ConvexMasterChef.sol#276)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-0.6/utils/Address.sol#119)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accCvxPerShare).div(1e12) (contracts/convex-platform/ConvexMasterChef.sol#277)
Reentrancy in ConvexMasterChef.deposit(uint256,uint256) (contracts/convex-platform/ConvexMasterChef.sol#215-242):
        External calls:
        - safeRewardTransfer(msg.sender,pending) (contracts/convex-platform/ConvexMasterChef.sol#225)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-0.6/token/ERC20/SafeERC20.sol#69)
                - cvx.safeTransfer(_to,cvxBal) (contracts/convex-platform/ConvexMasterChef.sol#308)
                - cvx.safeTransfer(_to,_amount) (contracts/convex-platform/ConvexMasterChef.sol#310)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-0.6/utils/Address.sol#119)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (contracts/convex-platform/ConvexMasterChef.sol#227-231)
        External calls sending eth:
        - safeRewardTransfer(msg.sender,pending) (contracts/convex-platform/ConvexMasterChef.sol#225)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-0.6/utils/Address.sol#119)
        State variables written after the call(s):
        - user.amount = user.amount.add(_amount) (contracts/convex-platform/ConvexMasterChef.sol#232)
        - user.rewardDebt = user.amount.mul(pool.accCvxPerShare).div(1e12) (contracts/convex-platform/ConvexMasterChef.sol#233)
Reentrancy in ConvexMasterChef.withdraw(uint256,uint256) (contracts/convex-platform/ConvexMasterChef.sol#245-266):
        External calls:
        - safeRewardTransfer(msg.sender,pending) (contracts/convex-platform/ConvexMasterChef.sol#253)
                - returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (node_modules/@openzeppelin/contracts-0.6/token/ERC20/SafeERC20.sol#69)
                - cvx.safeTransfer(_to,cvxBal) (contracts/convex-platform/ConvexMasterChef.sol#308)
                - cvx.safeTransfer(_to,_amount) (contracts/convex-platform/ConvexMasterChef.sol#310)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-0.6/utils/Address.sol#119)
        External calls sending eth:
        - safeRewardTransfer(msg.sender,pending) (contracts/convex-platform/ConvexMasterChef.sol#253)
                - (success,returndata) = target.call{value: value}(data) (node_modules/@openzeppelin/contracts-0.6/utils/Address.sol#119)
        State variables written after the call(s):
        - user.amount = user.amount.sub(_amount) (contracts/convex-platform/ConvexMasterChef.sol#254)
        - user.rewardDebt = user.amount.mul(pool.accCvxPerShare).div(1e12) (contracts/convex-platform/ConvexMasterChef.sol#255)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

ConvexMasterChef.pendingCvx(uint256,address) (contracts/convex-platform/ConvexMasterChef.sol#158-181) performs a multiplication on the result of a division:
        -cvxReward = multiplier.mul(rewardPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/convex-platform/ConvexMasterChef.sol#172-175)
        -accCvxPerShare = accCvxPerShare.add(cvxReward.mul(1e12).div(lpSupply)) (contracts/convex-platform/ConvexMasterChef.sol#176-178)
ConvexMasterChef.updatePool(uint256) (contracts/convex-platform/ConvexMasterChef.sol#192-212) performs a multiplication on the result of a division:
        -cvxReward = multiplier.mul(rewardPerBlock).mul(pool.allocPoint).div(totalAllocPoint) (contracts/convex-platform/ConvexMasterChef.sol#203-206)
        -pool.accCvxPerShare = pool.accCvxPerShare.add(cvxReward.mul(1e12).div(lpSupply)) (contracts/convex-platform/ConvexMasterChef.sol#208-210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

ConvexMasterChef.updatePool(uint256) (contracts/convex-platform/ConvexMasterChef.sol#192-212) uses a dangerous strict equality:
        - lpSupply == 0 (contracts/convex-platform/ConvexMasterChef.sol#198)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in ConvexMasterChef.emergencyWithdraw(uint256) (contracts/convex-platform/ConvexMasterChef.sol#289-302):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (contracts/convex-platform/ConvexMasterChef.sol#292)
        State variables written after the call(s):
        - user.amount = 0 (contracts/convex-platform/ConvexMasterChef.sol#294)
        - user.rewardDebt = 0 (contracts/convex-platform/ConvexMasterChef.sol#295)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

convex-platform/CrvDepositor.sol

```
Reentrancy in CrvDepositor.lockCurve() (contracts/convex-platform/CrvDepositor.sol#140-149):
        External calls:
        - _lockCurve() (contracts/convex-platform/CrvDepositor.sol#142)
                - 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(crvBpt).safeTransfer(staker,crvBalance) (contracts/convex-platform/CrvDepositor.sol#114)
                - IStaker(staker).increaseAmount(crvBalanceStaker) (contracts/convex-platform/CrvDepositor.sol#124)
                - IStaker(staker).increaseTime(unlockAt) (contracts/convex-platform/CrvDepositor.sol#132)
        - ITokenMinter(minter).mint(msg.sender,incentiveCrv) (contracts/convex-platform/CrvDepositor.sol#146)
        External calls sending eth:
        - _lockCurve() (contracts/convex-platform/CrvDepositor.sol#142)
                - (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-platform/CrvDepositor.sol#147)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

CrvDepositor.initialLock() (contracts/convex-platform/CrvDepositor.sol#88-104) performs a multiplication on the result of a division:
        -unlockInWeeks = (unlockAt / WEEK) * WEEK (contracts/convex-platform/CrvDepositor.sol#95)
CrvDepositor._lockCurve() (contracts/convex-platform/CrvDepositor.sol#107-135) performs a multiplication on the result of a division:
        -unlockInWeeks = (unlockAt / WEEK) * WEEK (contracts/convex-platform/CrvDepositor.sol#128)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

CrvDepositor._lockCurve() (contracts/convex-platform/CrvDepositor.sol#107-135) uses a dangerous strict equality:
        - crvBalanceStaker == 0 (contracts/convex-platform/CrvDepositor.sol#119)
CrvDepositor.initialLock() (contracts/convex-platform/CrvDepositor.sol#88-104) uses a dangerous strict equality:
        - vecrv == 0 (contracts/convex-platform/CrvDepositor.sol#93)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in CrvDepositor._lockCurve() (contracts/convex-platform/CrvDepositor.sol#107-135):
        External calls:
        - IERC20(crvBpt).safeTransfer(staker,crvBalance) (contracts/convex-platform/CrvDepositor.sol#114)
        - IStaker(staker).increaseAmount(crvBalanceStaker) (contracts/convex-platform/CrvDepositor.sol#124)
        - IStaker(staker).increaseTime(unlockAt) (contracts/convex-platform/CrvDepositor.sol#132)
        State variables written after the call(s):
        - unlockTime = unlockInWeeks (contracts/convex-platform/CrvDepositor.sol#133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

CrvDepositor.setFees(uint256) (contracts/convex-platform/CrvDepositor.sol#72-78) contains a tautology or contradiction:
        - _lockIncentive >= 0 && _lockIncentive <= 30 (contracts/convex-platform/CrvDepositor.sol#75)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction

CrvDepositor.initialLock() (contracts/convex-platform/CrvDepositor.sol#88-104) ignores return value by IStaker(staker).release() (contracts/convex-platform/CrvDepositor.sol#98)
CrvDepositor.initialLock() (contracts/convex-platform/CrvDepositor.sol#88-104) ignores return value by IStaker(staker).createLock(crvBalanceStaker,unlockAt) (contracts/convex-platform/CrvDepositor.sol#101)
CrvDepositor._lockCurve() (contracts/convex-platform/CrvDepositor.sol#107-135) ignores return value by IStaker(staker).increaseAmount(crvBalanceStaker) (contracts/convex-platform/CrvDepositor.sol#124)
CrvDepositor._lockCurve() (contracts/convex-platform/CrvDepositor.sol#107-135) ignores return value by IStaker(staker).increaseTime(unlockAt) (contracts/convex-platform/CrvDepositor.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

convex-platform/ExtraRewardStashV3.sol

```
Reentrancy in ExtraRewardStashV3.claimRewards() (contracts/convex-platform/ExtraRewardStashV3.sol#96-120):
        External calls:
        - checkForNewRewardTokens() (contracts/convex-platform/ExtraRewardStashV3.sol#100)
                - rewardContract = IRewardFactory(rewardFactory).CreateTokenRewards(_token,mainRewardContract,address(this)) (contracts/convex-platform/ExtraRewardStashV3.sol#170-173)
        - IDeposit(operator).setGaugeRedirect(pid) (contracts/convex-platform/ExtraRewardStashV3.sol#104)
        State variables written after the call(s):
        - hasRedirected = true (contracts/convex-platform/ExtraRewardStashV3.sol#105)
Reentrancy in ExtraRewardStashV3.setToken(address) (contracts/convex-platform/ExtraRewardStashV3.sol#159-180):
        External calls:
        - rewardContract = IRewardFactory(rewardFactory).CreateTokenRewards(_token,mainRewardContract,address(this)) (contracts/convex-platform/ExtraRewardStashV3.sol#170-173)
        State variables written after the call(s):
        - t.rewardAddress = rewardContract (contracts/convex-platform/ExtraRewardStashV3.sol#175)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

ExtraRewardStashV3.claimRewards() (contracts/convex-platform/ExtraRewardStashV3.sol#96-120) ignores return value by IDeposit(operator).setGaugeRedirect(pid) (contracts/convex-platform/ExtraRewardStashV3.sol#104)
ExtraRewardStashV3.claimRewards() (contracts/convex-platform/ExtraRewardStashV3.sol#96-120) ignores return value by IDeposit(operator).claimRewards(pid,gauge) (contracts/convex-platform/ExtraRewardStashV3.sol#111)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

convex-platform/PoolManagerSecondaryProxy.sol

```
PoolManagerSecondaryProxy.shutdownPool(uint256) (contracts/convex-platform/PoolManagerSecondaryProxy.sol#84-98) ignores return value by IPools(pools).shutdownPool(_pid) (contracts/convex-platform/
PoolManagerSecondaryProxy.sol#91)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

convex-platform/PoolManagerV3.sol

```
PoolManagerV3._addPool(address,uint256) (contracts/convex-platform/PoolManagerV3.sol#69-78) ignores return value by IPools(pools).addPool(lptoken,_gauge,_stashVersion) (contracts/convex-platform/P
oolManagerV3.sol#77)
PoolManagerV3.shutdownPool(uint256) (contracts/convex-platform/PoolManagerV3.sol#87-92) ignores return value by IPools(pools).shutdownPool(_pid) (contracts/convex-platform/PoolManagerV3.sol#90)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

convex-platform/VoteProxy.sol

```
VoterProxy.claimFees(address,address) (contracts/convex-platform/VoterProxy.sol#333-339) ignores return value by IFeeDistributor(_distroContract).claimToken(address(this),_token) (contracts/convex-platform/VoterProxy.sol#335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

- No major issues were found by Slither.
- All the reentrancy vulnerabilities were checked individually, and they are all false positives.
- The multiplications on the result of divisions are intentional or have minimal impact.
- Unchecked transfers were correctly flagged by Sither, although it makes no sense to check the return value in this case, as any failed transfer would revert directly.

4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

aura-contracts/AuraBalRewardPool.sol

Line	SWC Title	Severity	Short Description
78	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
184	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
184	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
185	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
186	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

aura-contracts/AuraMath.sol

Line	SWC Title	Severity	Short Description
15	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
19	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
23	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
27	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
36	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
36	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "%" discovered
36	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
68	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
75	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
79	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
86	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered

aura-contracts/AuraMerkleDrop.sol

Line	SWC Title	Severity	Short Description
74	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
77	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
106	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
119	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
155	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
155	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
156	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
157	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered

aura-contracts/AuraStakingProxy.sol

Line	SWC Title	Severity	Short Description
22	(SWC-123) Requirement Violation	Low	Requirement violation.
151	(SWC-123) Requirement Violation	Low	Requirement violation.

aura-contracts/AuraVestedEscrow.sol

Line	SWC Title	Severity	Short Description
67	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
105	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
106	(SWC-110) Assert Violation	Unknown	Out of bounds array access
108	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
108	(SWC-110) Assert Violation	Unknown	Out of bounds array access
109	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
111	(SWC-110) Assert Violation	Unknown	Out of bounds array access
145	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
154	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
167	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
168	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
168	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
187	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered

aura-contracts/BalInvestor.sol

Line	SWC Title	Severity	Short Description
50	(SWC-110) Assert Violation	Unknown	Out of bounds array access
51	(SWC-110) Assert Violation	Unknown	Out of bounds array access
52	(SWC-110) Assert Violation	Unknown	Out of bounds array access
55	(SWC-110) Assert Violation	Unknown	Out of bounds array access
64	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
64	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
71	(SWC-110) Assert Violation	Unknown	Out of bounds array access
72	(SWC-110) Assert Violation	Unknown	Out of bounds array access
74	(SWC-110) Assert Violation	Unknown	Out of bounds array access
75	(SWC-110) Assert Violation	Unknown	Out of bounds array access

aura-contracts/RewardPoolDepositWrapper.sol

Line	SWC Title	Severity	Short Description
45	(SWC-113) DoS with Failed Call	Medium	Multiple calls are executed in the same transaction.

convex-platform/BaseRewardPool.sol

Line	SWC Title	Severity	Short Description
84	(SWC-110) Assert Violation	Unknown	Public state variable with array type causing reacheable exception by default.
218	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
219	(SWC-110) Assert Violation	Unknown	Out of bounds array access
234	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
235	(SWC-110) Assert Violation	Unknown	Out of bounds array access
266	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
267	(SWC-110) Assert Violation	Unknown	Out of bounds array access
300	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
301	(SWC-110) Assert Violation	Unknown	Out of bounds array access
353	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered

convex-platform/ConvexMasterChef.sol

Line	SWC Title	Severity	Short Description
111	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
112	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
167	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
170	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
194	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
199	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
202	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
211	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
293	(SWC-107) Reentrancy	Low	Read of persistent state following external call.
298	(SWC-107) Reentrancy	Low	Read of persistent state following external call.

convex-platform/PoolManagerV3.sol

Line	SWC Title	Severity	Short Description
16	(SWC-123) Requirement Violation	Low	Requirement violation.
74	(SWC-123) Requirement Violation	Low	Requirement violation.

convex-platform/ProxyFactory.sol

Line	SWC Title	Severity	Short Description
24	(SWC-104) Unchecked Call Return Value	Medium	Unchecked return value from external call.

convex-platform/VoterProxy.sol

Line	SWC Title	Severity	Short Description
17	(SWC-123) Requirement Violation	Low	Requirement violation.
257	(SWC-123) Requirement Violation	Low	Requirement violation.

- No major issues were found by MythX.
- The requirement violations and assert violations are all false positives.
- Integer Overflows and Underflows flagged by MythX are false positives.
- `block.number` is not used as a source of randomness in any of the smart contracts.
- DoS with Failed Call was correctly flagged by MythX, although the likelihood is minimal.

THANK YOU FOR CHOOSING

// HALBORN