// HALBORN

CENTAURSWAP

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **May 8th-16th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	05/08/2021	Nishit Majithia
0.2	Document Edits	05/11/2021	Ferran Celades
1.0	Final Document	05/16/2021	Nishit Majithia
1.1	Remediation Plan	05/31/2021	Nishit Majithia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Ferran Celades	Halborn	ferran.celades@halborn.com
Nishit Majithia	Halborn	nishit.majithia@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Centaur Swap uses single-side staking to provide liquidity. In single-side staking, a liquidity provider only needs to stake a single asset rather than in the pair-based format that is seen on popular AMMs, where an LP needs to stake a pair of tokens in equal value to provide liquidity to a pool.

Centaur Swap is designed to allow individual asset pools to trade with each other to maximise liquidity utilisation. This means that once an asset pool is funded, it can effectively be paired against every other existing asset pool.

Centaur Swap engaged Halborn to conduct a security assessment on their Smart contracts beginning on May 8th, 2021. The security assessment was scoped to the smart contract provided in the Github repository Centaur Smart Contracts and an audit of the security risk and implications regarding the changes introduced by the development team at Centaur Swap prior to its production release shortly following the assessments deadline.

Though this security audit's outcome is satisfactory, only the most essential aspects were tested and verified to achieve objectives and deliverables set in the scope due to time and resource constraints. It is essential to note the use of the best practices for secure smart-contract development.

1.2 AUDIT SUMMARY

The team at Halborn was provided one week for the engagement and assigned two full time security engineers to audit the security of the smart contract. The security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit to achieve the following:

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

In summary, Halborn identified few security risks, and recommends performing further testing to validate extended safety and correctness in context to the whole set of contracts. External threats, such as economic attacks, oracle attacks, and inter-contract functions and calls should be validated for expected logic and state.

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 the smart contract audit. While manual testing is recommended to uncover flaws in logic, process,and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of 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 (brownie console and manual deployments on Ganache)
- Manual testing by custom Python 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 (Goerli, 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. It's quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was 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.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW

3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

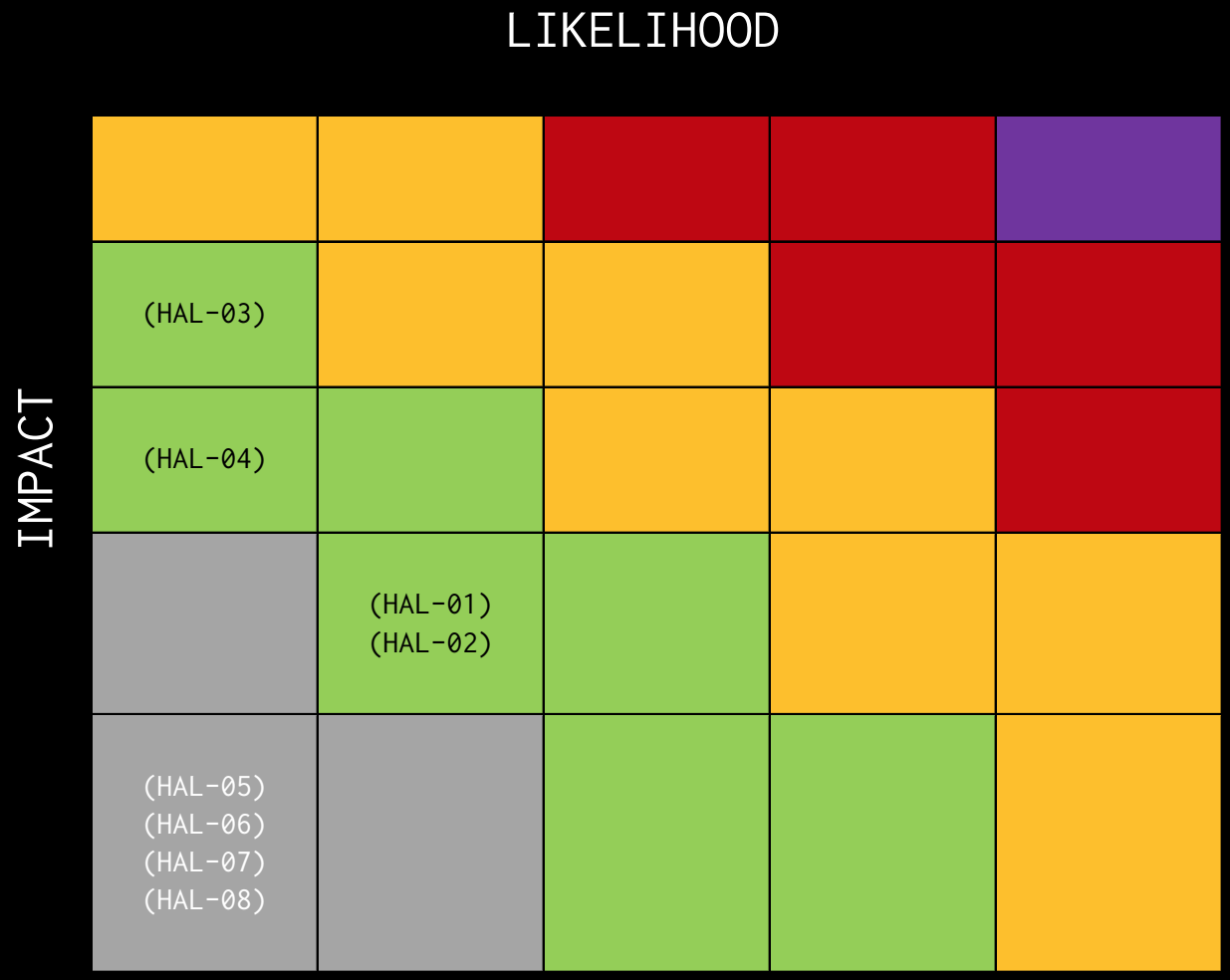
IN-SCOPE:
The security assessment was scoped to the smart contracts:
- CentaurFactory.sol
- CentaurLPToken.sol
- CentaurPool.sol
- CentaurRouter.sol
- CentaurSettlement.sol
- WheyFarm.sol
- WheyToken.sol
- All smart contracts under helpers, interfaces and libraries folders.

Commit ID: 697348e14359fe49cc84a56c5d7fc54cd9ed5260

OUT-OF-SCOPE:
External libraries and economics attacks.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	4	4



EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
INFINITE ALLOWANCE	Low	SOLVED - 05/31/2021
INTEGER OVERFLOW	Low	RISK ACCEPTED
USE OF BLOCK.TIMESTAMP	Low	RISK ACCEPTED
FLOATING PRAGMA AND VERSION MISMATCH	Low	RISK ACCEPTED
POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	RISK ACCEPTED
FUNCTION REDEFINITION	Informational	RISK ACCEPTED
USER CONTROLLED REQUIRE COMPARISON	Informational	SOLVED - 05/31/2021
MISSING VARIABLE CHECKS	Informational	RISK ACCEPTED
MANUAL TESTING	-	-
STATIC ANALYSIS	-	-
AUTOMATED SECURITY SCAN	-	-

FINDINGS & TECH DETAILS

3.1 (HAL-01) INFINITE ALLOWANCE - LOW

Description:

Setting the allowance value to -1 or MAX UINT 256 on the CentaurLPToken contract will cause the spender to keep performing transfers until a new approval is set on the spender.

Code Location:

Listing 1: CentaurLPToken.sol (Lines 54)

```
53     function transferFrom(address from, address to, uint value)
           external returns (bool) {
54         if (allowance[from][msg.sender] != uint(-1)) {
55             allowance[from][msg.sender] = allowance[from][msg.
                   sender].sub(value);
56         }
57         _transfer(from, to, value);
58         return true;
59     }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendations:

Follow the standard ERC20 practices and allow the user to increase and decrease the approval amount (see increaseApproval and decreaseApproval of StandardToken.sol#L63-L98).

If this is not possible, ensure users are aware of this extra functionality and encourage them to use it when appropriate. Furthermore, it is

preferable to periodically increase the allowance rather than disabling the allowance feature.

Remediation Plan:

SOLVED: Centaur team has updated the document regarding this functionality here https://docs.cntr.finance/audits

3.2 (HAL-02) INTEGER OVERFLOW - LOW

Description:

A subtraction underflow may occur when calling transferFrom on the CentaurLPToken contract. For example, if the allowance of the spender is 0, the value will be subtracted from it, causing an underflow. See Figure 1.

Integer overflows or underflows occur when the result of an arithmetic operation is outside of the possible range for an integer. If the amount exceeds the maximum or is lower than the minimum represented by the number of bits available, it will result in an incorrect value.

```
Transaction sent: 0xa07b35e87712a035a50cdc2f90d7560dd4065528362e9cb3fce68f9975fdd28d
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 9
  CentaurRouter.removeLiquidity confirmed (SafeMath: subtraction overflow) - Block: 10   Gas used:
33251 (0.28%)
```

Figure 1: Underflow taking place on the transferFrom function in the CentaurLPToken due to allowance being 0.

Code Location:

Listing 2: CentaurLPToken.sol (Lines 55)

```
53     function transferFrom(address from, address to, uint value)
           external returns (bool) {
54         if (allowance[from][msg.sender] != uint(-1)) {
55             allowance[from][msg.sender] = allowance[from][msg.
                   sender].sub(value);
56         }
57         _transfer(from, to, value);
58         return true;
59     }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendations:

Although safe math libraries are used, no message is returned to the caller when the underflow causes the transaction to fail. Returning an error message like "ERC20: transfer amount exceeds allowance" would help explain why the transaction failed.

Reference:

Ethereum Smart Contract Best Practices - Integer Overflow and Underflow

Remediation Plan:

RISK ACCEPTED: Centaur team accepted this risk since the contracts are already deployed.

3.3 (HAL-03) USE OF BLOCK.TIMESTAMP - LOW

Description:

The global variable block.timestamp does not necessarily hold the current time, and may not be accurate. Miners can influence the value of block.timestamp to perform Maximal Extractable Value (MEV) attacks. There is no guarantee that the value is correct, only that it is higher than the previous block's timestamp.

Code Location:

Listing 3: CentaurSettlement.sol (Lines 65)

```
60      require(msg.sender == _tPool, 'CentaurSwap: INVALID_POOL');
61
62      require(pendingSettlement[_sender][_fPool].settlementTimestamp
            != 0, 'CentaurSwap: SETTLEMENT_DOES_NOT_EXISTS');
63      require(pendingSettlement[_sender][_tPool].settlementTimestamp
            != 0, 'CentaurSwap: SETTLEMENT_DOES_NOT_EXISTS');
64
65      require(block.timestamp >= pendingSettlement[_sender][_fPool].
            settlementTimestamp, 'CentaurSwap: SETTLEMENT_PENDING');
66
67      _safeTransfer(ICentaurPool(_tPool).baseToken(), _tPool,
            pendingSettlement[_sender][_fPool].maxAmountOut);
```

Listing 4: CentaurRouter.sol (Lines 23)

```
22      modifier ensure(uint deadline) {
23          require(deadline >= block.timestamp, 'CentaurSwap: EXPIRED
                ');
24          _;
```

Listing 5: CentaurPool.sol (Lines 161)

```
149 ICentaurSettlement.Settlement memory pendingSettlement =
        ICentaurSettlement.Settlement(
150         pool,
151         _amountIn,
152         ICentaurPool(pool).baseTokenTargetAmount(),
153         (ICentaurPool(pool).baseTokenBalance()).sub(_amountIn),
154         ICentaurPool(pool).liquidityParameter(),
155         address(this),
156         maxAmount,
157         baseTokenTargetAmount,
158         baseTokenBalance,
159         liquidityParameter,
160         _receiver,
161         block.timestamp.add(ICentaurSettlement(settlement).
                settlementDuration())
162     );
```

Listing 6: CentaurPool.sol (Lines 201)

```
199 require (pendingSettlement.settlementTimestamp != 0, 'CentaurSwap:
        NO_PENDING_SETTLEMENT');
200 require (pendingSettlement.tPool == address(this), 'CentaurSwap:
        WRONG_POOL_SETTLEMENT');
201 require (block.timestamp >= pendingSettlement.settlementTimestamp,
        'CentaurSwap: SETTLEMENT_STILL_PENDING');
```

Listing 7: CentaurPool.sol (Lines 306)

```
303 function emergencyWithdraw(address _token, uint _amount, address
        _to) external onlyFactory {
304     _safeTransfer(_token, _to, _amount);
305
306     emit EmergencyWithdraw(block.timestamp, _token, _amount, _to);
307 }
```

Recommendation:

Use `block.number` instead of `block.timestamp` to reduce the risk of MEV attacks. If possible, use an oracle.

Remediation Plan:

RISK ACCEPTED: Centaur team accepted this risk since the contracts are already deployed.

3.4 (HAL-04) FLOATING PRAGMA AND VERSION MISMATCH - LOW

Description:

Some contracts are using a floating pragma, such as ^0.6.12. Deploy contracts with the same compiler version and flags used during development and testing. Locking the pragma helps ensure that contracts do not accidentally get deployed using a different compiler specification. For example, an outdated compiler version might introduce bugs, or a new version that is not extensively tested may introduce security vulnerabilities.

Additionally, the contracts are using different versions of the solidity compiler. Listing # 8 shows the versions used.

Listing 8: (Lines 1)

```
1  - Version used: ['=0.6.12', '>=0.4.24<0.8.0', '>=0.5.0',
       '>=0.6.0', '>=0.6.0<0.8.0', '>=0.6.2', '>=0.6.2<0.8.0',
       '^0.5.0||^0.6.0||^0.7.0', '^0.6.12']
2  - =0.6.12 (contracts/CentaurFactory.sol#3)
3  - =0.6.12 (contracts/CentaurLPToken.sol#3)
4  - =0.6.12 (contracts/CentaurPool.sol#3)
5  - ABIEncoderV2 (contracts/CentaurPool.sol#4)
6  - =0.6.12 (contracts/CentaurRouter.sol#3)
7  - =0.6.12 (contracts/CentaurSettlement.sol#3)
8  - ABIEncoderV2 (contracts/CentaurSettlement.sol#4)
9  - >=0.5.0 (contracts/interfaces/ICentaurFactory.sol#3)
10 - >=0.5.0 (contracts/interfaces/ICentaurPool.sol#3)
11 - >=0.6.2 (contracts/interfaces/ICentaurRouter.sol#3)
12 - >=0.5.0 (contracts/interfaces/ICentaurSettlement.sol#3)
13 - ABIEncoderV2 (contracts/interfaces/ICentaurSettlement.sol#4)
14 - ^0.6.12 (contracts/interfaces/ICloneFactory.sol#3)
```

```
15 - >=0.6.0<0.8.0 (contracts/interfaces/IERC20.sol#3)
16 - ^0.6.12 (contracts/interfaces/IOracle.sol#3)
17 - >=0.5.0 (contracts/interfaces/IWETH.sol#3)
18 - ^0.5.0||^0.6.0||^0.7.0 (contracts/libraries/ABDKMathQuad.sol#6)
19 - ^0.6.12 (contracts/libraries/CentaurMath.sol#3)
20 - >=0.6.0<0.8.0 (contracts/libraries/Context.sol#3)
21 - >=0.4.24<0.8.0 (contracts/libraries/Initializable.sol#4)
22 - >=0.6.0<0.8.0 (contracts/libraries/Ownable.sol#3)
23 - >=0.6.0<0.8.0 (contracts/libraries/SafeMath.sol#3)
24 - >=0.6.0 (contracts/libraries/TransferHelper.sol#3)
```

Code Location:

Listing 9: interfaces/ICloneFactory.sol

```
3 pragma solidity ^0.6.12;
```

Listing 10: libraries/ABDKMathQuad.sol

```
6 pragma solidity ^0.5.0 || ^0.6.0 || ^0.7.0;
```

Listing 11: interfaces/IOracle.sol

```
3 pragma solidity ^0.6.12;
```

Listing 12: libraries/CentaurMath.sol

```
3 pragma solidity ^0.6.12;
```

Listing 13: helpers/CloneFactory.sol

```
3 pragma solidity ^0.6.12;
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendations:

Lock the pragma version whenever possible and avoid using a floating pragma in the final deployment. The pragma can be locked in the code by removing the caret (^) and by specifying the exact version in the Truffle configuration file truffle-config.js or hardhat.config.js if using the HardHat framework.

Remediation Plan:

RISK ACCEPTED: Centaur team accepted this risk since the contracts are already deployed.

3.5 (HAL-05) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

Public functions consume more gas than those declared as external. The EVM passes the arguments of a public function via pointers, which require the array to be in memory. There is no such requirement for external functions. Choosing the incorrect modifier can inflate the needed gas.

As explained by the Solidity Documentation:

- External functions ... can be called from other contracts and via transactions. An external function cannot be called internally
- Public functions ... can be either called internally or via messages.
- Internal functions ... can only be accessed internally (i.e. from within the current contract or contracts deriving from it)
- Private functions ... are only visible for the contract they are defined in and not in derived contracts.

Code Location:

Listing 14: WheyFarm.sol

```
92     function add(
93         uint256 _allocPoint,
94         IERC20 _lpToken,
95         bool _withUpdate
96     ) public onlyOwner {
```

Listing 15: WheyFarm.sol

```
127    function set(
128        uint256 _pid,
129        uint256 _allocPoint,
130        bool _withUpdate
131    ) public onlyOwner {
```

Listing 16: WheyFarm.sol

```
261    function deposit(uint256 _pid, uint256 _amount) public {
```

Listing 17: WheyFarm.sol

```
288    function withdraw(uint256 _pid, uint256 _amount) public {
```

Listing 18: WheyFarm.sol

```
308    function harvestAll() public {
```

Listing 19: WheyFarm.sol

```
327    function emergencyWithdraw(uint256 _pid) public {
```

Listing 20: WheyFarm.sol

```
346    function updateDev(address _devaddr) onlyOwner public {
```

Listing 21: WheyToken.sol

```
10     function mint(address _to, uint256 _amount) public onlyOwner {
```

Recommendation:

If appropriate, declare the functions as 'external' instead of 'public'. A best practice is to use external if expecting a function only to be called externally, public if called internally and externally, and private or internal if only used inside the contract. Public functions are always accessible, but external functions are only available to outside callers.

Remediation Plan:

RISK ACCEPTED: Centaur team accepted this risk since the contracts are already deployed.

3.6 (HAL-06) FUNCTION REDEFINITION - INFORMATIONAL

Description:

The function _safeTransfer is re-declared in CentaurPool and CentaurSettlement. Other contracts are using the declared safeTransfer on the TransferHelper library contract.

Code Location:

Listing 22: CentaurPool.sol

```
94 function _safeTransfer(address token, address to, uint value)
       private {
95     (bool success, bytes memory data) = token.call(abi.
           encodeWithSelector(SELECTOR, to, value));
```

```
96     require(success && (data.length == 0 || abi.decode(data, (bool
           ))), 'CentaurSwap: TRANSFER_FAILED');
97 }
```

Listing 23: CentaurSettlement.sol

```
33 function _safeTransfer(address token, address to, uint value)
       private {
34     (bool success, bytes memory data) = token.call(abi.
           encodeWithSelector(SELECTOR, to, value));
35     require(success && (data.length == 0 || abi.decode(data, (bool
           ))), 'CentaurSwap: TRANSFER_FAILED');
36 }
```

Recommendation:

Apply library usage consistently to avoid security issues. If using a library for transfers, it is better to use it everywhere. Patching will only have to be done for one instance rather than multiple if updates are required.

Remediation Plan:

RISK ACCEPTED: Centaur team accepted this risk since the contracts are already deployed.

3.7 (HAL-07) USER CONTROLLED REQUIRE COMPARISON - INFORMATIONAL

Description:

The `_minLiquidity` variable in the `addLiquidity` function of the `CentaurRouter` contract is used in a `require` comparison. The user can control both ends of the `require` comparison. For example, setting the `_minLiquidity` value to `0` would allow a user to bypass the `require` check

and allow adding any amount of liquidity.

Code Location:

Listing 24: CentaurRouter.sol (Lines 84)

```
70 function addLiquidity(
71     address _baseToken,
72     uint _amount,
73     address _to,
74     uint _minLiquidity,
75     uint _deadline
76 ) external virtual override ensure(_deadline) onlyEOA(msg.sender)
      returns (uint amount, uint liquidity) {
77     address pool = ICentaurFactory(factory).getPool(_baseToken);
78     require(pool != address(0), 'CentaurSwap: POOL_NOT_FOUND');
79
80     (liquidity) = _addLiquidity(_baseToken, _amount, _minLiquidity
          );
81
82     TransferHelper.safeTransferFrom(_baseToken, msg.sender, pool,
          _amount);
83     liquidity = ICentaurPool(pool).mint(_to);
84     require(liquidity > _minLiquidity, 'CentaurSwap:
          INSUFFICIENT_OUTPUT_AMOUNT');
85
86     return (_amount, liquidity);
87 }
```

Listing 25: CentaurRouter.sol (Lines 67)

```
50 function _addLiquidity(
51     address _baseToken,
52     uint _amount,
53     uint _minLiquidity
54 ) internal view virtual returns (uint liquidity) {
55     ICentaurPool pool = ICentaurPool(ICentaurFactory(factory).
          getPool(_baseToken));
56
57     uint _totalSupply = pool.totalSupply();
58     uint _baseTokenTargetAmount = pool.baseTokenTargetAmount();
59     liquidity = _amount;
60
```

```
61      if (_totalSupply == 0) {
62          liquidity = _amount.add(_baseTokenTargetAmount);
63      } else {
64          liquidity = _amount.mul(_totalSupply).div(
                _baseTokenTargetAmount);
65      }
66
67      require(liquidity > _minLiquidity, 'CentaurSwap:
            INSUFFICIENT_OUTPUT_AMOUNT');
68  }
```

Recommendation:

Add the minimum liquidity amount during the construction of the pool. If this value needs to be changed later, adding a setter would allow updating its value. It is not a good practice to use user-controlled parameters for assertion checks in the code.

Remediation Plan:

SOLVED: Centaur team has updated the document regarding this functionality here https://docs.cntr.finance/audits. `_minLiquidity` is used for slippage tolerance therefore should be modifiable by the user.

3.8 (HAL-08) MISSING VARIABLE CHECKS - INFORMATIONAL

Description:

The value `_allocPoint` provided as argument on the `add` function in `WheyFarm` is not checked for being bigger than `0`. Setting a `totalAllocPoint` of `0` causes division-by-zero errors as shown in Figure 13.

```
>>> router.removeLiquidity(pool1.baseToken(), 1, accounts[2], 0, int(time.time()) + 2, {'from':deplo
yer})
Transaction sent: 0x85b660f04133d89977f859f26dc63fb16b61da007365b13fb67b4218a2c120f1
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 32
  CentaurRouter.removeLiquidity confirmed (SafeMath: subtraction overflow) - Block: 12418447   Gas u
sed: 33239 (0.28%)

<Transaction '0x85b660f04133d89977f859f26dc63fb16b61da007365b13fb67b4218a2c120f1'>
>>> pool1.approve(router, 100, {'from':deployer})
Transaction sent: 0x0d68dceb24ea30982a8ca7e0275e0705af300554321859ed2f75aeca9c508471
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 33
  CentaurPool.approve confirmed - Block: 12418448   Gas used: 29876 (0.25%)

<Transaction '0x0d68dceb24ea30982a8ca7e0275e0705af300554321859ed2f75aeca9c508471'>
>>> router.removeLiquidity(pool1.baseToken(), 100, accounts[2], 0, int(time.time()) + 2, {'from':dep
loyer})
Transaction sent: 0x9d51fbb0ee44e23ddf040af827baa015e38a94bad51633b710af397471e2e57a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 34
  CentaurRouter.removeLiquidity confirmed - Block: 12418449   Gas used: 61938 (0.52%)

<Transaction '0x9d51fbb0ee44e23ddf040af827baa015e38a94bad51633b710af397471e2e57a'>
>>>
```

.

Code Location:

Listing 26: WheyFarm.sol (Lines 102)

```
92 function add(
93     uint256 _allocPoint,
94     IERC20 _lpToken,
95     bool _withUpdate
96 ) public onlyOwner {
97     if (_withUpdate) {
98         massUpdatePools();
99     }
100    uint256 lastRewardBlock =
101        block.number > startBlock ? block.number : startBlock;
102    totalAllocPoint = totalAllocPoint.add(_allocPoint);
```

Listing 27: WheyFarm.sol (Lines 188)

```
186 if (block.number > pool.lastRewardBlock && lpSupply != 0) {
187     uint256 wheyReward = getWheyReward(pool.nextEmissionIndex,
            pool.lastRewardBlock, block.number);
188     uint256 poolWheyReward = wheyReward.mul(pool.allocPoint).div(
            totalAllocPoint);
189
190     accWheyPerShare = accWheyPerShare.add(
191         (poolWheyReward.mul(85).div(100)).mul(1e12).div(lpSupply)
192     );
193 }
```

Recommendation:

Perform boundary checks on user/admin supplied values to address possible issues.

Remediation Plan:

RISK ACCEPTED: Centaur team accepted this risk since the contracts are already deployed.

MANUAL TESTING

During the manual testing multiple questions where considered while evaluation each of the defined functions:

- Can it be re-called changing admin/roles and permissions?
- Can somehow an external controlled contract call again the function during the execution of it? (Re-entrancy)
- Can it be called twice in the same block and cause issues?
- Do we control sensitive or vulnerable parameters?
- Does the function check for boundaries on the parameters and internal values? Bigger than zero or equal? Argument count, array sizes, integer truncation . . .
- Are the function parameters and variables controlled by external contracts?
- Can extended contracts cause issues on the extender contract?

4.1 CentaurFactory

Attempting to add an empty pool does require the token address to be non-zero, ensuring that the pool was initialized as shown in Figure 2.

```
>>> CentaurPool[1].baseToken()
'0x0000000000000000000000000000000000000000'
>>> factory.addPool(CentaurPool[1])
Transaction sent: 0xce402709c4a793c185cf88cc997bd272c39af939f1cd514f224bf6d30ad126d2
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 7
  CentaurFactory.addPool confirmed (CentaurSwap: ZERO_ADDRESS) - Block: 8   Gas used: 25516 (0.21%)

<Transaction '0xce402709c4a793c185cf88cc997bd272c39af939f1cd514f224bf6d30ad126d2'>
>>>
```

Figure 2: Adding an empty pool on the Factory checks

Initializing a pool with a custom base token and later adding it to the factory is allowed as expected as seen in Figure 3.

Re-adding to the pool checks shown in Figure 4.

Removing the pool and trying double remove, shown in Figure 5.

Creating a pool using exported methods, shown in Figure 6.

```
>>> pool1.init(factory, AAVE_MAINNET, AAVE_ETH_ORACLE, web3.toWei(100, 'ether') )
Transaction sent: 0xebd2b759fe1a5978b4ad4b9c16f47ed7f7c3cdeb775911ea2f8cf0684d38d10e
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 7
  CentaurPool.init confirmed - Block: 12412878   Gas used: 234578 (1.95%)

<Transaction '0xebd2b759fe1a5978b4ad4b9c16f47ed7f7c3cdeb775911ea2f8cf0684d38d10e'>
>>> factory.addPool(pool1)
Transaction sent: 0x6c1e720f8c3121c35cc9810f790006d60b5614814cf0422556c53e611bed26a8
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 8
  CentaurFactory.addPool confirmed - Block: 12412879   Gas used: 88929 (0.74%)

<Transaction '0x6c1e720f8c3121c35cc9810f790006d60b5614814cf0422556c53e611bed26a8'>
>>>
```

Figure 3: Initializing the pool and adding it to the factory

```
>>> factory.addPool(pool1)
Transaction sent: 0x79482c688976253175ded9e07e84a3e5934da4982d81e9f013394b25dea38fc5
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 16
  CentaurFactory.addPool confirmed (CentaurSwap: POOL_EXISTS) - Block: 12412887   Gas used: 26429 (0
.22%)

<Transaction '0x79482c688976253175ded9e07e84a3e5934da4982d81e9f013394b25dea38fc5'>
>>>
```

Figure 4: Trying to re-add the pool to the factory

```
>>> factory.removePool(AAVE_MAINNET)
Transaction sent: 0x8347a5eeb774dcdb9ea8df8109b3e9bf4c6e2fb3dc9ecfd8abea6edfca46435a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 9
  CentaurFactory.removePool confirmed (reverted) - Block: 12412880   Gas used: 27153 (0.23%)

<Transaction '0x8347a5eeb774dcdb9ea8df8109b3e9bf4c6e2fb3dc9ecfd8abea6edfca46435a'>
>>> factory.removePool(pool1)
Transaction sent: 0xd6afc393b9c6b42e6c0bac8527e084366132a95566662265fd4e5e23c4f80a39
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 10
  CentaurFactory.removePool confirmed - Block: 12412881   Gas used: 25444 (0.21%)

<Transaction '0xd6afc393b9c6b42e6c0bac8527e084366132a95566662265fd4e5e23c4f80a39'>
>>> factory.removePool(pool1)
Transaction sent: 0xdadf03ceb9fbe76e63474f7896239cc3feb0a86e76621ae621f1141a43a9cc06
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 11
  CentaurFactory.removePool confirmed (CentaurSwap: POOL_NOT_FOUND) - Block: 12412882   Gas used: 26
444 (0.22%)

<Transaction '0xdadf03ceb9fbe76e63474f7896239cc3feb0a86e76621ae621f1141a43a9cc06'>
>>>
```

Figure 5: Trying to double remove the pool

```
>>> factory.createPool(aave, AAVE_ETH_ORACLE, 0)
Transaction sent: 0x8a3b72125462443c670146b8961d6d43b24c79e991390d3f788cce55f6d06f34
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 7
  CentaurFactory.createPool confirmed - Block: 12413923   Gas used: 343084 (2.86%)

<Transaction '0x8a3b72125462443c670146b8961d6d43b24c79e991390d3f788cce55f6d06f34'>
>>> factory.allPools(1)
  File "<console>", line 1, in <module>
  File "brownie/network/contract.py", line 1712, in __call__
    return self.call(*args, block_identifier=block_identifier)
  File "brownie/network/contract.py", line 1517, in call
    raise VirtualMachineError(e) from None
VirtualMachineError: invalid opcode: invalid opcode
>>> factory.allPools(0)
'0x521629cbe068e58b43f1aEaB73E47fC43231E67C'
>>> factory.allPoolsLength()
1
```

Figure 6: Creating a pool using methods

4.2 CentaurPool

The internal function named _safeTransfer duplicates the safeTransfer function in TransferHelper.

Re-initializing the factory causes errors, as shown in Figure 7.

```
>>> pool1.init(factory, AAVE_MAINNET, AAVE_ETH_ORACLE, web3.toWei(100, 'ether') )
Transaction sent: 0x96d04a6fd7a0130e1dbc54985486aa5281415c24532d4fce8d9448a50d241ce3
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 17
  CentaurPool.init confirmed (Initializable: contract is already initialized) - Block: 12412888   Ga
s used: 25528 (0.21%)

<Transaction '0x96d04a6fd7a0130e1dbc54985486aa5281415c24532d4fce8d9448a50d241ce3'>
>>>
```

Figure 7: Factory re-initialization checks

Some functions are required to be called from the Router:

- mint
- burn
- swapFrom
- swapTo

Functions allowed by the Factory:

- setFactory

- setTradeEnabled
- setDepositEnabled
- setWithdrawEnabled
- setLiquidityParameter
- emergencyWithdraw

No requirement from the caller:

- swapSettle
- mint assumes that the balance increase is a result of the addLiquidity call from the Router:
- baseTokenBalance keeps track of the previous balance value, it subtracts to the current balanceOf base token.

Test showcasing that if funds are transferred the next account gets accredited the minted tokens is shown in Figure 8 and Figure 9.

When swapping tokens CentaurMath is calculating the difference of surplus and demand using the function shown in their documentation https://docs.cntr.finance/centaur-swap/cross-pair-swaps??.

```
router = CentaurRouter.at(factory.router())
factory.createPool(aave, AAVE_ETH_ORACLE, 0)

pool1 = CentaurPool.at(factory.allPools(0))

# Enable deposit
factory.setPoolDepositEnabled(pool1, True)

# Approve router to spend deployer aave tokens
aave.approve(router, 1000, {'from': deployer})

# Imagine that user transfers funds to the pool without calling addliquidity
aave.transfer(pool1, 5000, {'from': user})

# The user calling `addLiquidity` will be accredited with the mint of the
# transfered tokens plus the one he/she provides
router.addLiquidity(aave, 1000, deployer, 0, int(time.time()), {'from':deployer})

# pool1.balanceOf(deployer) == 6000
# >>> True
```

Figure 8: Test case that demonstrated that manually transferred funds to the pool get minted by the following user adding liquidity

```
Transaction sent: 0x87d53c4c59596f8c9c105feb50b0f7ed455cd40c60916302abba38eb3cc5f6ab
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 63
  ERC20Mock.approve confirmed - Block: 12413982   Gas used: 44023 (0.37%)

Transaction sent: 0x751c20c6ade831aa6cb887b231ec0548960d39af83add2ee71e1a0d421880ec2
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 2
  ERC20Mock.transfer confirmed - Block: 12413983   Gas used: 51151 (0.43%)

Transaction sent: 0xf0eb6d8ac9c158dff3b4be4438cf2896a5d5e34ff987d3f6f57c3180db12170e
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 64
  CentaurRouter.addLiquidity confirmed - Block: 12413984   Gas used: 137091 (1.14%)

>>> pool1.balanceOf(deployer)
6000
>>>
```

Figure 9: Result of the testcase on the console

4.3 CentaurRouter

The removeLiquidity does not work if the client does not approve the router from spending the underlying CentaurLPTokens, furthermore the subtraction overflow is shown indicating that the transferFrom function is missing to subtract the allowance.

```
>>> farm.poolInfo(0)
("0xe0aA552A10d7EC8760Fc6c246D391E698a82dDf9", 0, 12425383, 0, 100, 2)
>>> 12425383 - 12425378
5
>>> farm.updatePool(0)
Transaction sent: 0x758456711ce816f7fa3fd7e5b06263df25ce4f8ec044e16356f5c409ec45e5d0
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 13
  WheyFarm.updatePool confirmed (SafeMath: division by zero) - Block: 12425384   Gas used: 31385 (0.
26%)

<Transaction '0x758456711ce816f7fa3fd7e5b06263df25ce4f8ec044e16356f5c409ec45e5d0'>
>>>
```

Figure 10: Subtraction overflow on the allowance due to invalid checks. Missing throw error description

Anti-contract call protection modifier onlyEOA(msg.sender) is present and used on multiple places, only whitelisted contracts are allowed to interact with it:

Listing 28

```
1 modifier onlyEOA(address _address) {
2         if (onlyEOAEnabled) {
3             require((!Address.isContract(_address) ||
                  whitelistContracts[_address]), 'CentaurSwap:
                  ONLY_EOA_ALLOWED');
4         }
5         _;
6     }
```

4.4 Settlements

It does not validate that the caller is the pool, so a user can addSettlement and removeSettlement.

It is not possible to remove settlements that are pending on removeSettlement, which would allow a bypass of the settlement system:

Listing 29

```
require(block.timestamp >= pendingSettlement[_sender][_fPool].
    settlementTimestamp, 'CentaurSwap: SETTLEMENT_PENDING');
```

It does not allow modifying a settlement during the addSettlement function, which would allow decreasing the settlement time or even invalidating it:

Listing 30

```
require(pendingSettlement[_sender][_pendingSettlement.fPool].
    settlementTimestamp == 0, 'CentaurSwap: SETTLEMENT_EXISTS');
require(pendingSettlement[_sender][_pendingSettlement.tPool].
    settlementTimestamp == 0, 'CentaurSwap: SETTLEMENT_EXISTS');
```

4.5 Helpers

- safeTransferETH is used at the end of the functions after updating the status

4.6 WheyFarm

- 400 WHEY per block during the early-adopter period of 16th April 2021 to 19th April 2021
- 100 WHEY per block from 19th April 2021 to 16th May 2021
- Reduction of 10 WHEY per block monthly for the next eight months
- Reduction of 10% every month until 16th May 2024
- 1 WHEY per block until 16th August 2026

The scheme is implemented in scripts/emissionPerEpoch.json, the output is shown in Figure 11.

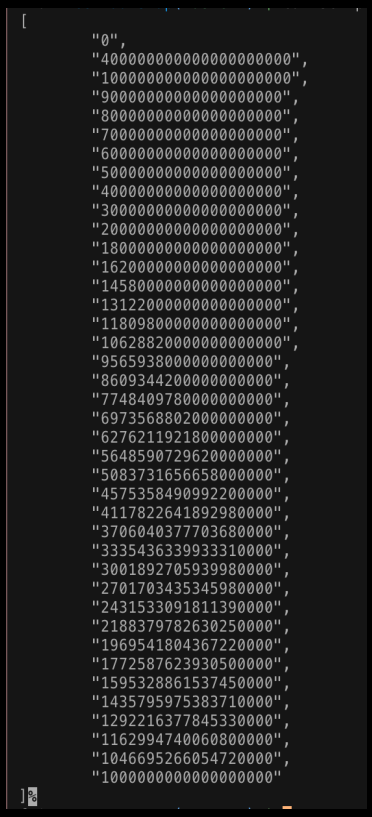


Figure 11: Emission used as described on the documentation

Trying a deposit successfully updates the balance on the internal structs as shown in Figure 12.

- totalAllocPoint should be required to be greater than zero or a division by zero error may be triggered, as shown in Figure 13.

Listing 31

```
1 WheyFarm.pendingWhey(uint256,address) (contracts/WheyFarm.sol
    #177-195) performs a multiplication on the result of a division
    :
2    -accWheyPerShare = accWheyPerShare.add((poolWheyReward.mul(85)
        .div(100)).mul(1e12).div(lpSupply)) (contracts/WheyFarm.sol
        #190-192)
3 WheyFarm.updatePool(uint256) (contracts/WheyFarm.sol#206-258)
    performs a multiplication on the result of a division:
4    -poolWheyReward = wheyReward.mul(pool.allocPoint).div(
        totalAllocPoint) (contracts/WheyFarm.sol#232)
5    -devReward = poolWheyReward.mul(15).div(100) (contracts/
        WheyFarm.sol#235)
```

```
<Transaction '0x69813465d0c846b32c7309c1e0ae86c92967c13cfb7ad5e4d60ee0b158d843d7'>
>>> farm.deposit(0, 100)
Transaction sent: 0x33ee7e9004d32b76e3073ddc4ba48066a87c0ecfcadcd470ae00e7b8bbc0091a
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 12
  WheyFarm.deposit confirmed - Block: 12425383   Gas used: 133523 (1.11%)

<Transaction '0x33ee7e9004d32b76e3073ddc4ba48066a87c0ecfcadcd470ae00e7b8bbc0091a'>
>>> weth.balanceOf(farm)
100
>>> weth.balanceOf(deployer)
999999999999999999999900
>>> farm.poolInfo
<ContractCall 'poolInfo(uint256)'>
>>> farm.poolInfo(0)
("0xe0aA552A10d7EC8760Fc6c246D391E698a82dDf9", 0, 12425383, 0, 100, 2)
>>> farm.userInfo
<ContractCall 'userInfo(uint256, address)'>
>>> farm.userInfo(deployer)
  File "<console>", line 1, in <module>
  File "brownie/network/contract.py", line 1712, in __call__
    return self.call(*args, block_identifier=block_identifier)
  File "brownie/network/contract.py", line 1512, in call
    tx.update({"to": self._address, "data": self.encode_input(*args)})
  File "brownie/network/contract.py", line 1593, in encode_input
    data = format_input(self.abi, args)
  File "brownie/convert/normalize.py", line 20, in format_input
    raise type(e)(f"{abi['name']} {e}") from None
ValueError: userInfo Sequence has incorrect length, expected 2 but got 1
>>> farm.userInfo(0,deployer)
(100, 0)
>>> farm.poolInfo(0)
("0xe0aA552A10d7EC8760Fc6c246D391E698a82dDf9", 0, 12425383, 0, 100, 2)
>>>
```

Figure 12: Successfully updating internal structs

```
>>> farm.poolInfo(0)
("0xe0aA552A10d7EC8760Fc6c246D391E698a82dDf9", 0, 12425383, 0, 100, 2)
>>> 12425383 - 12425378
5
>>> farm.updatePool(0)
Transaction sent: 0x758456711ce816f7fa3fd7e5b06263df25ce4f8ec044e16356f5c409ec45e5d0
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 13
  WheyFarm.updatePool confirmed (SafeMath: division by zero) - Block: 12425384   Gas used: 31385 (0.
26%)

<Transaction '0x758456711ce816f7fa3fd7e5b06263df25ce4f8ec044e16356f5c409ec45e5d0'>
>>>
```

Figure 13: Missing bigger than 0 check on totalAllocPoint is causing division by zero errors

```
6 Reference: https://github.com/crytic/slither/wiki/Detector-
    Documentation#divide-before-multiply
```

devs are exactly 15%, as shown in Figure 14.

```
>>> (1 - (whey.balanceOf(farm) / ((whey.balanceOf(deployer) - web3.toWei(250000,'ether')) + whey.balanceOf(farm)))) * 100
15.000000000000002
>>>
```

Figure 14: Math operation demonstrating that the percentage given to the devs is 15%

It correctly calculates the block reward based on block numbers, init and end values, shown in Figure 15.

```
>>> farm.getWheyReward(0,12425519,12425518)
  File "<console>", line 1, in <module>
  File "brownie/network/contract.py", line 1712, in __call__
    return self.call(*args, block_identifier=block_identifier)
  File "brownie/network/contract.py", line 1517, in call
    raise VirtualMachineError(e) from None
VirtualMachineError: revert: WheyFarm: _to less than _from
>>> farm.getWheyReward(0,12425519,12425525)
640000000000000000000
>>> farm.getWheyReward(0,12425518,12425525)
640000000000000000000
>>> farm.getWheyReward(0,12425518,12425520)
400000000000000000000
>>> farm.getWheyReward(0,12425516,12425520)
400000000000000000000
>>> farm.getWheyReward(0,12425516,12425524)
640000000000000000000
>>> farm.getWheyReward(0,12425516,12425527)
640000000000000000000
>>>
```

Figure 15: Rewarded whey based on init and end block numbers

```
>>> whey.balanceOf(farm)
4250000000000000000
>>> whey.balanceOf(user)
50150000000000000000
>>> whey.balanceOf(deployer)
250096000000000000000000
>>> whey.balanceOf(deployer) - web3.toWei(250000,'ether') + whey.balanceOf(user) + whey.balanceOf(farm)
64000000000000000000
>>> farm.withdraw(0, 1, {'from':user})
Transaction sent: 0x2841a41b94f2294b4e0dd40783c7dfb24d7d121dec231c0cbfec4c1ae3c2d403
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 44
  WheyFarm.withdraw confirmed - Block: 12425699   Gas used: 170890 (1.42%)

<Transaction '0x2841a41b94f2294b4e0dd40783c7dfb24d7d121dec231c0cbfec4c1ae3c2d403'>
>>> whey.balanceOf(deployer) - web3.toWei(250000,'ether') + whey.balanceOf(user) + whey.balanceOf(farm)
65000000000000000000
>>> farm.pendingWhey(0, user)
0
>>> whey.balanceOf(farm)
1
>>> whey.balanceOf(user)
552499999999999999999
>>> whey.balanceOf(deployer)
250097500000000000000000
>>> farm.withdraw(0, 4, {'from':user})
Transaction sent: 0x0dbffe2fca95755348ff9f532e5f8a6fd2708b41de96be03639e32ef1d68e819
  Gas price: 0.0 gwei   Gas limit: 12000000   Nonce: 45
  WheyFarm.withdraw confirmed - Block: 12425700   Gas used: 106061 (0.88%)

<Transaction '0x0dbffe2fca95755348ff9f532e5f8a6fd2708b41de96be03639e32ef1d68e819'>
>>> whey.balanceOf(deployer) - web3.toWei(250000,'ether') + whey.balanceOf(user) + whey.balanceOf(farm)
65000000000000000000
```

Figure 16: Correctly calculating withdraw amounts based on internal values and fee percentages

AUTOMATED TESTING

5.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped contract. 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. 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.

```
INFO:Detectors:
CentaurFactory (contracts/CentaurFactory.sol#14-179) contract sets array length with a user-controlled value:
        - allPools.push(pool) (contracts/CentaurFactory.sol#72)
CentaurFactory (contracts/CentaurFactory.sol#14-179) contract sets array length with a user-controlled value:
        - allPools.pop() (contracts/CentaurFactory.sol#95)
CentaurFactory (contracts/CentaurFactory.sol#14-179) contract sets array length with a user-controlled value:
        - allPools.push(_pool) (contracts/CentaurFactory.sol#83)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment
```

```
INFO:Detectors:
WheyFarm (contracts/WheyFarm.sol#12-367) contract sets array length with a user-controlled value:
        - poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0,0,nextEmissionIndex)) (contracts/WheyFarm.sol#114-123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment
```

- All array length assignments issues reported by the tool here are false positives.

```
INFO:Detectors:
CentaurPool.mint(address) (contracts/CentaurPool.sol#99-116) uses a dangerous strict equality:
        - totalSupply == 0 (contracts/CentaurPool.sol#103)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in CentaurPool.burn(address) (contracts/CentaurPool.sol#118-134):
        External calls:
        - _safeTransfer(baseToken,to,amount) (contracts/CentaurPool.sol#128)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/CentaurPool.sol#95)
        State variables written after the call(s):
        - baseTokenBalance = baseTokenBalance.sub(amount) (contracts/CentaurPool.sol#130)
        - baseTokenTargetAmount = baseTokenTargetAmount.sub(amount) (contracts/CentaurPool.sol#131)
Reentrancy in CentaurFactory.createPool(address,address,uint256) (contracts/CentaurFactory.sol#59-75):
        External calls:
        - pool = ICloneFactory(cloneFactory).createClone(poolLogic) (contracts/CentaurFactory.sol#63)
        - ICentaurPool(pool).init(address(this),_baseToken,_oracle,_liquidityParameter) (contracts/CentaurFactory.sol#64-69)
        State variables written after the call(s):
        - getPool[_baseToken] = pool (contracts/CentaurFactory.sol#71)
Reentrancy in CentaurSettlement.removeSettlement(address,address,address) (contracts/CentaurSettlement.sol#55-71):
        External calls:
        - _safeTransfer(ICentaurPool(_tPool).baseToken(),_tPool,pendingSettlement[_sender][_fPool].maxAmountOut) (contracts/CentaurSettlement.sol#67)
                - (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,value)) (contracts/CentaurSettlement.sol#34)
        State variables written after the call(s):
        - delete pendingSettlement[_sender][_fPool] (contracts/CentaurSettlement.sol#69)
        - delete pendingSettlement[_sender][_tPool] (contracts/CentaurSettlement.sol#70)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
CentaurRouter.removeLiquidity(address,uint256,address,uint256,uint256) (contracts/CentaurRouter.sol#108-123) ignores return value by ICentaurPool(pool).transferF
rom(msg.sender,pool,_liquidity) (contracts/CentaurRouter.sol#118)
CentaurRouter.swapExactTokensForTokens(address,uint256,address,uint256,address,uint256) (contracts/CentaurRouter.sol#145-161) ignores return value by ICentaurPoo
l(outputTokenPool).swapTo(msg.sender,_fromToken,finalAmountIn,value,_to) (contracts/CentaurRouter.sol#160)
CentaurRouter.swapExactETHForTokens(address,uint256,address,uint256) (contracts/CentaurRouter.sol#163-178) ignores return value by ICentaurPool(outputTokenPool).
swapTo(msg.sender,WETH,finalAmountIn,value,_to) (contracts/CentaurRouter.sol#177)
CentaurRouter.swapTokensForExactTokens(address,uint256,address,uint256,address,uint256) (contracts/CentaurRouter.sol#180-197) ignores return value by ICentaurPoo
l(outputTokenPool).swapTo(msg.sender,_fromToken,finalAmountIn,value,_to) (contracts/CentaurRouter.sol#196)
CentaurRouter.swapETHForExactTokens(address,uint256,address,uint256) (contracts/CentaurRouter.sol#199-217) ignores return value by ICentaurPool(outputTokenPool).
swapTo(msg.sender,WETH,finalAmountIn,value,_to) (contracts/CentaurRouter.sol#214)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

```
INFO:Detectors:
WheyFarm.pendingWhey(uint256,address) (contracts/WheyFarm.sol#177-195) performs a multiplication on the result of a division:
        -accWheyPerShare = accWheyPerShare.add((poolWheyReward.mul(85).div(100)).mul(1e12).div(lpSupply)) (contracts/WheyFarm.sol#190-192)
WheyFarm.updatePool(uint256) (contracts/WheyFarm.sol#206-258) performs a multiplication on the result of a division:
        -poolWheyReward = wheyReward.mul(pool.allocPoint).div(totalAllocPoint) (contracts/WheyFarm.sol#232)
        -devReward = poolWheyReward.mul(15).div(100) (contracts/WheyFarm.sol#235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
WheyFarm.updatePool(uint256) (contracts/WheyFarm.sol#206-258) uses a dangerous strict equality:
        - pool.lastRewardBlock > wheyEmissionSchedule[i].add(startBlock) && i == wheyEmissionSchedule.length - 1 (contracts/WheyFarm.sol#222)
WheyFarm.updatePool(uint256) (contracts/WheyFarm.sol#206-258) uses a dangerous strict equality:
        - pool.lastRewardBlock > wheyEmissionSchedule[i_scope_0].add(startBlock) && i_scope_0 == wheyEmissionSchedule.length - 1 (contracts/WheyFarm.sol#252)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
```

```
INFO:Detectors:
Reentrancy in WheyFarm.add(uint256,IERC20,bool) (contracts/WheyFarm.sol#92-124):
        External calls:
        - massUpdatePools() (contracts/WheyFarm.sol#98)
                - whey.mint(devaddr,devReward) (contracts/WheyFarm.sol#237)
                - whey.mint(address(this),poolReward) (contracts/WheyFarm.sol#238)
        State variables written after the call(s):
        - poolInfo.push(PoolInfo(_lpToken,_allocPoint,lastRewardBlock,0,0,nextEmissionIndex)) (contracts/WheyFarm.sol#114-123)
        - totalAllocPoint = totalAllocPoint.add(_allocPoint) (contracts/WheyFarm.sol#102)
Reentrancy in WheyFarm.deposit(uint256,uint256) (contracts/WheyFarm.sol#261-285):
        External calls:
        - updatePool(_pid) (contracts/WheyFarm.sol#264)
                - whey.mint(devaddr,devReward) (contracts/WheyFarm.sol#237)
                - whey.mint(address(this),poolReward) (contracts/WheyFarm.sol#238)
        - safeWheyTransfer(msg.sender,pending) (contracts/WheyFarm.sol#271)
                - whey.transfer(_to,wheyBal) (contracts/WheyFarm.sol#340)
                - whey.transfer(_to,_amount) (contracts/WheyFarm.sol#342)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (contracts/WheyFarm.sol#274-278)
        State variables written after the call(s):
        - pool.totalDeposit = pool.totalDeposit.add(_amount) (contracts/WheyFarm.sol#282)
        - user.amount = user.amount.add(_amount) (contracts/WheyFarm.sol#279)
        - user.rewardDebt = user.amount.mul(pool.accWheyPerShare).div(1e12) (contracts/WheyFarm.sol#280)
Reentrancy in WheyFarm.emergencyWithdraw(uint256) (contracts/WheyFarm.sol#327-334):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (contracts/WheyFarm.sol#330)
        State variables written after the call(s):
        - user.amount = 0 (contracts/WheyFarm.sol#332)
        - user.rewardDebt = 0 (contracts/WheyFarm.sol#333)
Reentrancy in WheyFarm.harvestAll() (contracts/WheyFarm.sol#308-324):
        External calls:
        - updatePool(i) (contracts/WheyFarm.sol#314)
                - whey.mint(devaddr,devReward) (contracts/WheyFarm.sol#237)
                - whey.mint(address(this),poolReward) (contracts/WheyFarm.sol#238)
        State variables written after the call(s):
        - user.rewardDebt = user.amount.mul(pool.accWheyPerShare).div(1e12) (contracts/WheyFarm.sol#319)
Reentrancy in WheyFarm.set(uint256,uint256,bool) (contracts/WheyFarm.sol#127-139):
        External calls:
        - massUpdatePools() (contracts/WheyFarm.sol#133)
                - whey.mint(devaddr,devReward) (contracts/WheyFarm.sol#237)
                - whey.mint(address(this),poolReward) (contracts/WheyFarm.sol#238)
        State variables written after the call(s):
        - poolInfo[_pid].allocPoint = _allocPoint (contracts/WheyFarm.sol#138)
        - totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint) (contracts/WheyFarm.sol#135-137)
Reentrancy in WheyFarm.updatePool(uint256) (contracts/WheyFarm.sol#206-258):
        External calls:
        - whey.mint(devaddr,devReward) (contracts/WheyFarm.sol#237)
        - whey.mint(address(this),poolReward) (contracts/WheyFarm.sol#238)
        State variables written after the call(s):
        - pool.accWheyPerShare = pool.accWheyPerShare.add(poolReward.mul(1e12).div(lpSupply)) (contracts/WheyFarm.sol#239-241)
        - pool.lastRewardBlock = block.number (contracts/WheyFarm.sol#243)
        - pool.nextEmissionIndex = i_scope_0 (contracts/WheyFarm.sol#248)
        - pool.nextEmissionIndex = i_scope_0 (contracts/WheyFarm.sol#253)
Reentrancy in WheyFarm.withdraw(uint256,uint256) (contracts/WheyFarm.sol#288-306):
        External calls:
        - updatePool(_pid) (contracts/WheyFarm.sol#292)
                - whey.mint(devaddr,devReward) (contracts/WheyFarm.sol#237)
                - whey.mint(address(this),poolReward) (contracts/WheyFarm.sol#238)
        - safeWheyTransfer(msg.sender,pending) (contracts/WheyFarm.sol#298)
                - whey.transfer(_to,wheyBal) (contracts/WheyFarm.sol#340)
                - whey.transfer(_to,_amount) (contracts/WheyFarm.sol#342)
        State variables written after the call(s):
        - user.amount = user.amount.sub(_amount) (contracts/WheyFarm.sol#300)
        - user.rewardDebt = user.amount.mul(pool.accWheyPerShare).div(1e12) (contracts/WheyFarm.sol#301)
Reentrancy in WheyFarm.withdraw(uint256,uint256) (contracts/WheyFarm.sol#288-306):
        External calls:
        - updatePool(_pid) (contracts/WheyFarm.sol#292)
                - whey.mint(devaddr,devReward) (contracts/WheyFarm.sol#237)
                - whey.mint(address(this),poolReward) (contracts/WheyFarm.sol#238)
        - safeWheyTransfer(msg.sender,pending) (contracts/WheyFarm.sol#298)
                - whey.transfer(_to,wheyBal) (contracts/WheyFarm.sol#340)
                - whey.transfer(_to,_amount) (contracts/WheyFarm.sol#342)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (contracts/WheyFarm.sol#302)
        State variables written after the call(s):
        - pool.totalDeposit = pool.totalDeposit.sub(_amount) (contracts/WheyFarm.sol#304)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
WheyFarm.safeWheyTransfer(address,uint256) (contracts/WheyFarm.sol#337-344) ignores return value by whey.transfer(_to,wheyBal) (contracts/WheyFarm.sol#340)
WheyFarm.safeWheyTransfer(address,uint256) (contracts/WheyFarm.sol#337-344) ignores return value by whey.transfer(_to,_amount) (contracts/WheyFarm.sol#342)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

- All unused return issues are present and it is good to add a return value check to avoid unexpected errors. Return value checks ensure proper exception handling.

- Re-entrancy issue is not present but in future to protect against cross-function reentrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the withdraw function with a recursive call. OpenZeppelin has it's own mutex implementation called ReentrancyGuard which provides a modifier to any function called nonReentrant that guards the function with a mutex against reentrancy attacks. In contracts CentaurFactory, CentaurPool, CentaurSettlement, WheyFarm this guard should be added.

```
INFO:Detectors:
CentaurFactory.constructor(address,address,address)._poolLogic (contracts/CentaurFactory.sol#28) lacks a zero-check on :
                - poolLogic = _poolLogic (contracts/CentaurFactory.sol#29)
CentaurFactory.constructor(address,address,address)._cloneFactory (contracts/CentaurFactory.sol#28) lacks a zero-check on :
                - cloneFactory = _cloneFactory (contracts/CentaurFactory.sol#30)
CentaurFactory.setPoolLogic(address)._poolLogic (contracts/CentaurFactory.sol#163) lacks a zero-check on :
                - poolLogic = _poolLogic (contracts/CentaurFactory.sol#164)
CentaurFactory.setCloneFactory(address)._cloneFactory (contracts/CentaurFactory.sol#167) lacks a zero-check on :
                - cloneFactory = _cloneFactory (contracts/CentaurFactory.sol#168)
CentaurFactory.setSettlement(address)._settlement (contracts/CentaurFactory.sol#171) lacks a zero-check on :
                - settlement = _settlement (contracts/CentaurFactory.sol#172)
CentaurFactory.setRouter(address)._router (contracts/CentaurFactory.sol#175) lacks a zero-check on :
                - router = _router (contracts/CentaurFactory.sol#176)
CentaurPool.init(address,address,address,uint256)._factory (contracts/CentaurPool.sol#75) lacks a zero-check on :
                - factory = _factory (contracts/CentaurPool.sol#76)
CentaurPool.init(address,address,address,uint256)._baseToken (contracts/CentaurPool.sol#75) lacks a zero-check on :
                - baseToken = _baseToken (contracts/CentaurPool.sol#77)
CentaurPool.init(address,address,address,uint256)._oracle (contracts/CentaurPool.sol#75) lacks a zero-check on :
                - oracle = _oracle (contracts/CentaurPool.sol#79)
CentaurPool.setFactory(address)._factory (contracts/CentaurPool.sol#283) lacks a zero-check on :
                - factory = _factory (contracts/CentaurPool.sol#284)
CentaurRouter.constructor(address,address)._factory (contracts/CentaurRouter.sol#39) lacks a zero-check on :
                - factory = _factory (contracts/CentaurRouter.sol#40)
CentaurRouter.constructor(address,address)._WETH (contracts/CentaurRouter.sol#39) lacks a zero-check on :
                - WETH = _WETH (contracts/CentaurRouter.sol#41)
CentaurRouter.setFactory(address)._factory (contracts/CentaurRouter.sol#276) lacks a zero-check on :
                - factory = _factory (contracts/CentaurRouter.sol#277)
CentaurSettlement.constructor(address,uint256)._factory (contracts/CentaurSettlement.sol#28) lacks a zero-check on :
                - factory = _factory (contracts/CentaurSettlement.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
add(uint256,IERC20,bool) should be declared external:
        - WheyFarm.add(uint256,IERC20,bool) (contracts/WheyFarm.sol#92-124)
set(uint256,uint256,bool) should be declared external:
        - WheyFarm.set(uint256,uint256,bool) (contracts/WheyFarm.sol#127-139)
deposit(uint256,uint256) should be declared external:
        - WheyFarm.deposit(uint256,uint256) (contracts/WheyFarm.sol#261-285)
withdraw(uint256,uint256) should be declared external:
        - WheyFarm.withdraw(uint256,uint256) (contracts/WheyFarm.sol#288-306)
harvestAll() should be declared external:
        - WheyFarm.harvestAll() (contracts/WheyFarm.sol#308-324)
emergencyWithdraw(uint256) should be declared external:
        - WheyFarm.emergencyWithdraw(uint256) (contracts/WheyFarm.sol#327-334)
updateDev(address) should be declared external:
        - WheyFarm.updateDev(address) (contracts/WheyFarm.sol#346-348)
getWheyPerBlock() should be declared external:
        - WheyFarm.getWheyPerBlock() (contracts/WheyFarm.sol#357-365)
mint(address,uint256) should be declared external:
        - WheyToken.mint(address,uint256) (contracts/WheyToken.sol#10-12)
owner() should be declared external:
        - Ownable.owner() (contracts/openzeppelin/contracts/access/Ownable.sol#35-37)
```

```
INFO:Detectors:
Reentrancy in WheyFarm.constructor(address,uint256,uint256[],uint256[]) (contracts/WheyFarm.sol#66-84):
        External calls:
        - whey.mint(_devaddr,25000000000000000000000) (contracts/WheyFarm.sol#75)
        State variables written after the call(s):
        - devaddr = _devaddr (contracts/WheyFarm.sol#77)
        - startBlock = _startBlock (contracts/WheyFarm.sol#78)
        - wheyEmissionPerEpoch = _wheyEmissionPerEpoch (contracts/WheyFarm.sol#82)
        - wheyEmissionSchedule = _wheyEmissionSchedule (contracts/WheyFarm.sol#81)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in WheyFarm.deposit(uint256,uint256) (contracts/WheyFarm.sol#261-285):
        External calls:
        - updatePool(_pid) (contracts/WheyFarm.sol#264)
                - whey.mint(devaddr,devReward) (contracts/WheyFarm.sol#237)
                - whey.mint(address(this),poolReward) (contracts/WheyFarm.sol#238)
        - safeWheyTransfer(msg.sender,pending) (contracts/WheyFarm.sol#271)
                - whey.transfer(_to,wheyBal) (contracts/WheyFarm.sol#340)
                - whey.transfer(_to,_amount) (contracts/WheyFarm.sol#342)
        - pool.lpToken.safeTransferFrom(address(msg.sender),address(this),_amount) (contracts/WheyFarm.sol#274-278)
        Event emitted after the call(s):
        - Deposit(msg.sender,_pid,_amount) (contracts/WheyFarm.sol#284)
Reentrancy in WheyFarm.emergencyWithdraw(uint256) (contracts/WheyFarm.sol#327-334):
        External calls:
        - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (contracts/WheyFarm.sol#330)
        Event emitted after the call(s):
        - EmergencyWithdraw(msg.sender,_pid,user.amount) (contracts/WheyFarm.sol#331)
Reentrancy in WheyFarm.withdraw(uint256,uint256) (contracts/WheyFarm.sol#288-306):
        External calls:
        - updatePool(_pid) (contracts/WheyFarm.sol#292)
                - whey.mint(devaddr,devReward) (contracts/WheyFarm.sol#237)
                - whey.mint(address(this),poolReward) (contracts/WheyFarm.sol#238)
        - safeWheyTransfer(msg.sender,pending) (contracts/WheyFarm.sol#298)
                - whey.transfer(_to,wheyBal) (contracts/WheyFarm.sol#340)
                - whey.transfer(_to,_amount) (contracts/WheyFarm.sol#342)
        - pool.lpToken.safeTransfer(address(msg.sender),_amount) (contracts/WheyFarm.sol#302)
        Event emitted after the call(s):
        - Withdraw(msg.sender,_pid,_amount) (contracts/WheyFarm.sol#305)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
WheyFarm.constructor(address,uint256,uint256[],uint256[])._devaddr (contracts/WheyFarm.sol#67) lacks a zero-check on :
                - devaddr = _devaddr (contracts/WheyFarm.sol#77)
WheyFarm.updateDev(address)._devaddr (contracts/WheyFarm.sol#346) lacks a zero-check on :
                - devaddr = _devaddr (contracts/WheyFarm.sol#347)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

- Issue regarding different pragma or floating pragma has been already mentioned in the above report.
- missing zero address check is issue in CentaurFactory, CentaurRouter contract. Add address validation for user-supplied values in addition to the existing OpenZeppelin RBAC controls.
- Use of block.timestamp issue is already mentioned in the above report.

5.2 AUTOMATED SECURITY SCAN

MYTHX:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruit 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 the testers machine and sent the compiled results to the analyzers to locate any vulnerabilities. Only security-related findings are shown below.

Results:

CentaurFactory.sol

Report for CentaurFactory.sol
https://dashboard.mythx.io/#/console/analyses/e5521208-099f-4788-a4eb-a282bf1c95ca

Line	SWC Title	Severity	Short Description
14	(SWC-123) Requirement Violation	Low	Requirement violation.
87	(SWC-123) Requirement Violation	Low	Requirement violation.
107	(SWC-123) Requirement Violation	Low	Requirement violation.
147	(SWC-123) Requirement Violation	Low	Requirement violation.

CentaurPool.sol

Report for CentaurPool.sol
https://dashboard.mythx.io/#/console/analyses/beb56831-286b-4d63-8c17-606fec3f5870

Line	SWC Title	Severity	Short Description
19	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
88	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
272	(SWC-000) Unknown	Medium	Function could be marked as external.
276	(SWC-000) Unknown	Medium	Function could be marked as external.

CentaurRouter.sol

Report for CentaurRouter.sol
https://dashboard.mythx.io/#/console/analyses/397416b9-6e3f-4dbd-a45e-fc524d85a10a

Line	SWC Title	Severity	Short Description
46	(SWC-110) Assert Violation	Low	An assertion violation was triggered.

CentaurSettlement.sol

Report for CentaurSettlement.sol
https://dashboard.mythx.io/#/console/analyses/5bcfc9e1-b4f8-4714-853c-a07c5375df44

Line	SWC Title	Severity	Short Description
15	(SWC-128) DoS With Block Gas Limit	Low	Potentially unbounded data structure passed to builtin.
21	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

WheyFarm.sol

Report for WheyFarm.sol
https://dashboard.mythx.io/#/console/analyses/500f71cd-1269-49ea-8061-af69631408c4

Line	SWC Title	Severity	Short Description
92	(SWC-000) Unknown	Medium	Function could be marked as external.
101	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
106	(SWC-128) DoS With Block Gas Limit	Medium	Loop over unbounded data structure.
127	(SWC-000) Unknown	Medium	Function could be marked as external.
154	(SWC-128) DoS With Block Gas Limit	Medium	Loop over unbounded data structure.
158	(SWC-128) DoS With Block Gas Limit	Medium	Loop over unbounded data structure.
186	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
187	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
200	(SWC-128) DoS With Block Gas Limit	Medium	Loop over unbounded data structure.
208	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
213	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
216	(SWC-128) DoS With Block Gas Limit	Medium	Loop over unbounded data structure.
231	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
243	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.
246	(SWC-128) DoS With Block Gas Limit	Medium	Loop over unbounded data structure.
261	(SWC-000) Unknown	Medium	Function could be marked as external.
288	(SWC-000) Unknown	Medium	Function could be marked as external.
308	(SWC-000) Unknown	Medium	Function could be marked as external.
310	(SWC-128) DoS With Block Gas Limit	Medium	Loop over unbounded data structure.
327	(SWC-000) Unknown	Medium	Function could be marked as external.
346	(SWC-000) Unknown	Medium	Function could be marked as external.
352	(SWC-128) DoS With Block Gas Limit	Medium	Implicit loop over unbounded data structure.
353	(SWC-128) DoS With Block Gas Limit	Medium	Implicit loop over unbounded data structure.
357	(SWC-000) Unknown	Medium	Function could be marked as external.
358	(SWC-128) DoS With Block Gas Limit	Low	Loop over unbounded data structure.
359	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

WheyToken.sol

```
Report for WheyToken.sol
https://dashboard.mythx.io/#/console/analyses/500f71cd-1269-49ea-8061-af69631408c4
```

Line	SWC Title	Severity	Short Description
10	(SWC-000) Unknown	Medium	Function could be marked as external.

library/CentaurMath.sol

```
Report for CentaurMath.sol
https://dashboard.mythx.io/#/console/analyses/1d518f69-e4bd-492f-bfb1-39b4e74a7363
```

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

THANK YOU FOR CHOOSING

// HALBORN