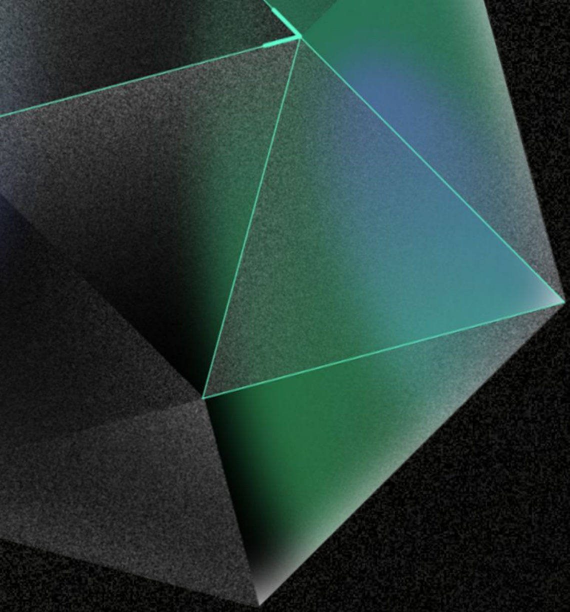
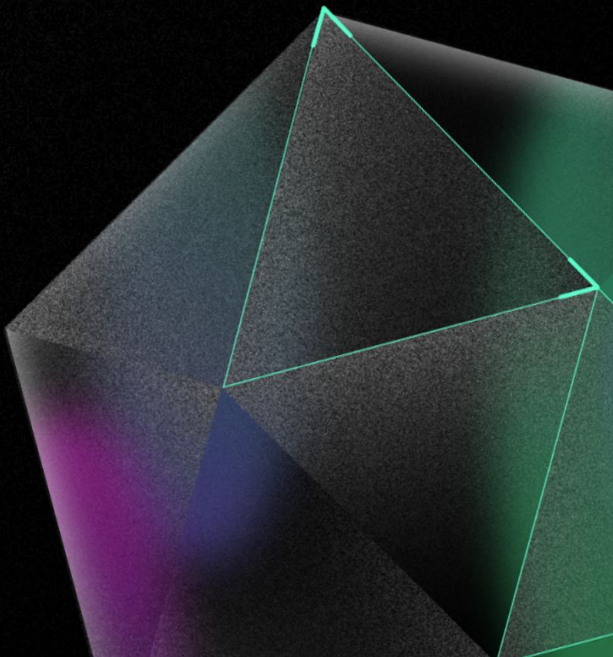


MANTISSA FINANCE SMART CONTRACT AUDIT REPORT

13.06.2022



Contents

Summary

Severity	Number of Findings
Critical	1
High	1
Medium	2
Low	4
Informational	3

Total: 11

Scope

The analyzed smart contracts are located in the following repository folder:
https://github.com/Mantissa-Finance/shared-audit/
(commit 6ef6a1cba15ac7cd6fd8e68b740c7cfe5b79d055)
The weaknesses described in this report were fixed in the following commit:
4780275ddff22b7a12a0e4fce34595ca21a612b5

Weaknesses

This section contains the list of discovered weaknesses.

1. Incorrect decimal calculation

Severity: Critical

Status: fixed

Remediation: multiply the first division parameter by 1e18, e.g. (fromPrice * 1e18) / toPrice <= priceChangeLimit

Description:

The function **checkSwapOraclePrice()** on line 535 in the file **Pool.sol** gets current prices for LP tokens from the oracle and calculates the price difference. The price difference is checked to be less or equal than **priceChangeLimit** which is represented with 18 decimals, although when dividing the two oracle prices the decimal precision is truncated and the result will always be less than **priceChangeLimit**. An illicit actor can leverage price spikes to swap tokens.

```
function checkSwapOraclePrice(ILP fromLp, ILP toLp) public view returns (bool
allowed) {
     uint256 fromPrice = tokenOraclePrice(address(fromLp));
     uint256 toPrice = tokenOraclePrice(address(toLp));
     if (fromPrice > toPrice) {
         allowed = fromPrice / toPrice <= priceChangeLimit;
     } else {
         allowed = toPrice / fromPrice <= priceChangeLimit;
     }
 }
```

2. Wrong zero amount check

Severity: High

Status: fixed

Remediation: change the modifier to use strict check amount > 0

Description:

The modifier **checkZeroAmount()** on line 80 in the file **Pool.sol** checks for the amount to be >= 0, thus allowing zero amount to pass.

```
modifier checkZeroAmount(uint256 amount) {
    require(amount >= 0, 'Amount cannot be 0');
    _;
 }
```

3. Incorrect swap token check

Severity: Medium

Status: fixed

Remediation: check for the corresponding LPs not to be equal (fromLp != toLp)

Description:

The function **swap()** checks that the input and output tokens are not the same on line 390 in the file **Pool.sol**, although it does not consider double entry point tokens such as **TUSD** (legacy vs new address), thus the check can be bypassed by an illicit actor and result in unexpected swap behaviour.

```
function swap(
      address from,
      address to,
      address recipient,
      uint256 amount,
      uint256 minAmount,
      uint256 deadline
  ) external whenNotPaused nonReentrant checkDeadline(deadline) checkZeroAmount(amount)
checkNullAddress(recipient) {
      require(from != to, "Cannot swap to same token");
...
      if (vars.treasuryFees > 0) {
         vars.toLp.withdrawUnderlyer(treasury, vars.treasuryFees);
      }
      vars.toLp.updateAsset(vars.toAmount + vars.treasuryFees, false);
      if (vars.lpAmount > 0) {
         vars.toLp.updateLiability(vars.lpAmount, true);
      }
      emit Swap(msg.sender, recipient, from, amount, to, vars.toAmount);
  }
```

4. Centralization issue

Severity: Medium

Status: acknowledged, multisig will be used

Remediation: implement a timelock mechanism or other centralization mitigation technique

Description:

The function **withdrawMnt()** on line 379 in the file **MasterMantis.sol** can be called by the owner, giving the ability to withdraw all the **MNT** tokens from the contract.

```
function withdrawMnt() external onlyOwner {
      mnt.safeTransfer(msg.sender, mnt.balanceOf(address(this)));
  }
}
```

5. Redundant check

Severity: Low

Status: fixed

Remediation: remove the checks

Description:

The checks on lines 61 and 74 in the file **LP.sol** are redundant since **Solidity** versions 0.8.4 and up have built-in underflow checks and the transaction will be reverted in case the condition is not met.

```
...
require(liability >= amount, "Cannot be negative");
...
require(currentAllowance >= amount, "Burn amount exceeds allowance.");
...
```

6. Redundant check

Severity: Low

Status: fixed

Remediation: remove the checks

Description:

The check on line 82 in the file **LP.sol** is redundant since the **SafeERC20** libarary's **safeTransfer()** function is used to transfer the tokens and it will revert the transaction in case the condition is not met.

```
require(underlier.balanceOf(address(this)) >= amount, "Not enough amount");
```

7. Redundant check

Severity: Low

Status: fixed

Remediation: remove the check

Description:

The check on line 42 in the file **MntNFT.sol** is redundant since the **for()** statement will not loop if **num == 0**.

```
function multiMint(uint256 num) external onlyOwner {
    require(num > 0, "Cannot be 0");
    for(uint i = 0; i < num; i++) {
        tokenCount++;
        _mint(msg.sender, tokenCount);
    }
}
```

8. Redundant check

Severity: Low

Status: fixed

Remediation: remove the check

Description:

The check on line 308 in the file **Pool.sol** is redundant since **Solidity** versions 0.8.4 and up have built-in underflow checks and the transaction will be reverted in case the condition is not met.

```
function withdraw(
      address token,
      address recipient,
      uint256 lpAmount,
...
      (uint256 amount, uint256 fees, uint256 treasuryFees) =
getWithdrawAmount(lpToken, lpAmount, false);
      require(amount > fees, "Fees too high");
      uint256 finalAmount = amount - fees;
      require(finalAmount >= minAmount, "Lower than minimum");
...
      lpToken.updateLiability(amount, false);
      emit Withdraw(msg.sender, recipient, token, lpAmount, finalAmount);
  }
```

9. Missing event

Severity: Informational

Status: fixed

Remediation: add the corresponding event

Description:

The function **setPool()** on line 36 in the file **LP.sol** updates the pool address but does not emit any event.

```
function setPool(address _pool) external onlyOwner {
    require(_pool != address(0), "Cannot be zero address");
    pool = _pool;
 }
```

10. Missing event

Severity: Informational

Status: fixed

Remediation: add the corresponding event

Description:

The functions **setFlashLoanParameters()** and **setLPFeed()** on lines 164 and 142 in the file **Pool.sol** update the flashloan parameters and price feed oracle address but do not emit any events.

```
function setFlashLoanParameters(uint256 _flashLimit, uint256 _flashFees)
external onlyOwner {
...
   }
function setLPFeed(address _token, address _feed) external
checkNullAddress(_token) onlyOwner {
...
   }
```

11. Incorrect error messages

Severity: Informational

Status: fixed

Remediation: change the error messages to correspond to the slippage parameter name

Description:

The **require** statements on lines 105 and 106 in the file **Pool.sol** in the function **setSlippageParams()** have incorrect error messages.

```
function setSlippageParams(uint256 _slippageA, uint256 _slippageN)
external onlyOwner {
    require(_slippageA > 0, "K cannot be 0");
    require(_slippageN > 0, "A cannot be 0");
    slippageA = _slippageA;
    slippageN = _slippageN;
  }
```

hexens