Math Master Audit Report

Version 1.0

palmcivet

June 24, 2024

Math Master Audit Report

palmcivet

June 24th, 2024

Prepared by: palmcivet

Table of Contents

    * [L-2] Incorrect function selector for `MathMasters::MathMasters__FullMulDivFailed()` custom error used in `MathMasters::mulWad` and `MathMasters::mulWadUp` functions
  - Informational
    * [I-1] Free memory pointer is being overwritten

Protocol Summary

This codebase was inspired by the solady, obront.eth, and solmate codebases. Huge thanks to karma for the help on FV with Halmos.

Disclaimer

The `palmcivet` team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
Likelihood	High	H	H/M	M
	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

- Commit Hash: c7643faa1a188a51b2167b68250816f90a9668c6

Scope

```
#-- MathMasters.sol
```

Roles

XX

Executive Summary

I am excited to begin learning about formal verification. The tools were enjoyable to use and the process was a rewarding challenge.

Issues found

Severity	Number of issues found
High	2
Medium	0
Low	2
Informational	1
Total	5

Findings

High

[H-1] `MathMasters::mulWadUp` function adds 1 to `x` in some situations, returning incorrect output

Description The `MathMasters::mulWadUp` function contains a line that, if certain conditions are met, adds 1 to `x`.

Impact This will return an incorrect result.

Proof of Concept

Code

Place the following code in the `MathMasters.t.sol` test file.

```
    function testMulWadUpUnit() public {
        uint256 x = 53438770891273403451;
        uint256 y = 53438770891273403445;
        uint256 result = MathMasters.mulWadUp(x, y);
        uint256 expected = x * y == 0 ? 0 : (x * y - 1) / 1e18 + 1;
        uint256 resultDown = MathMasters.mulWad(x, y);
        console2.log("result:", result); // 2855702234370009622372
        console2.log("expected:", expected); // 2855702234370009622319
        console2.log("resultDown:", resultDown); //
           2855702234370009622318
        assert(result != expected);
        assertEq(resultDown, expected - 1);
    }
```

Recommended Mitigation Remove this unnecessary line:

```
   function mulWadUp(uint256 x, uint256 y) internal pure returns (
      uint256 z) {
        /// @solidity memory-safe-assembly
        assembly {
            // Equivalent to `require(y == 0 || x <= type(uint256).max
               / y)`.
            if mul(
                y,
                gt(
                    x,
                    div(not(0), y)
                )
            ) {
                mstore(0x40, 0xbac65e5b)
                revert(0x1c, 0x04)
            }
-           if iszero(
-               sub(
-                   div(add(z, x), y), // is this zero? ((0 + x / y) -
    1)
-                   1
-               )
-           ) { x := add(x, 1) }
            z :=
                add(
                    iszero(
                        iszero(
                            mod(mul(x, y), WAD)
                        )
                    ),
```

```
28                      div(mul(x, y), WAD)
29                  )
30          }
31      }
```

[H-2] `MathMasters::sqrt` contains incorrect value when comparing to x to shift bits, returning incorrect output

Description In the `MathMasters::sqrt` function, assembly is used to find the square root `r` by shifting the bits left and right, comparing the outputs each time. However there is an incorrect value being used in a line of comparison. Each value used is a fullwidth hexadecimal (ie they consist soley of repeating `f`s), except for `16777002` - the hexadecimal value of which is `0xffff2a`.

```
1               // 87112285931760246646623899502532662132735 == 0
                    xffffffffffffffffffffffffffffffff
2               let r := shl(7, lt
                    (87112285931760246646623899502532662132735, x))
3               // 472236482869645213695 == 0xffffffffffffffff
4               r := or(r, shl(6, lt(472236482869645213695, shr(r, x))))
5               // 1099511627775 == 0xffffffffff
6               r := or(r, shl(5, lt(1099511627775, shr(r, x))))
7               // 16777002 == 0xffff2a
8               // 16777215 == 0xffffff
9               r := or(r, shl(4, lt(16777002, shr(r, x))))
```

Impact This will return the incorrect output.

Proof of Concept The `MathMasters::sqrt` function shares identical lines at the end of its implementation as its Solmate Equivalent. We can essentially remove these identical lines by creating new functions in our `Harness` contract with the differential sections of each `sqrt` function.

Code

Place the following code in the `./certora/harness/Harness.sol` file.

```
1  function solmateTopHalf(uint256 x) external pure returns (uint256 z) {
2          assembly {
3              let y := x
4
5              z := 181
6              if iszero(lt(y, 0x100000000000000000000000000000000)) {
7                  y := shr(128, y)
8                  z := shl(64, z)
9              }
10             if iszero(lt(y, 0x10000000000000000)) {
11                 y := shr(64, y)
12                 z := shl(32, z)
```

```
            }
            if iszero(lt(y, 0x10000000000)) {
                y := shr(32, y)
                z := shl(16, z)
            }
            if iszero(lt(y, 0x1000000)) {
                y := shr(16, y)
                z := shl(8, z)
            }
            z := shr(18, mul(z, add(y, 65536)))
        }
    }

    function mathMastersTopHalf(uint256 x) external pure returns (uint256 z) {
        assembly {
            z := 181

            let r := shl(7, lt(87112285931760246646623899502532662132735, x))
            r := or(r, shl(6, lt(4722366482869645213695, shr(r, x))))
            r := or(r, shl(5, lt(1099511627775, shr(r, x))))
            r := or(r, shl(4, lt(16777002, shr(r, x))))
            z := shl(shr(1, r), z)
            z := shr(18, mul(z, add(shr(r, x), 65536)))
        }
    }
```

Next we can use Halmos to compare these pieces of code.

Code

Place the following code in the `MathMasters.t.sol` test file.

```
function check_testHarnessWithCertoraEdgeCase(uint256 x) public {
        Harness harness = new Harness();
        assert(harness.mathMastersTopHalf(x) == harness.solmateTopHalf(x));
    }
```

This gives us multiple edge cases which we can demonstrate in a unit test.

Code

Place the following code in the `MathMasters.t.sol` test file.

```
function test_sqrt_breaks() public {
        // 24519788349697037967053404880661006174729938985413509120
        // 309481597173691432457732096
        // 72057151656296448
        //
            1053112873322795248289958139573627190922697279310508675642540687360
```

```
6           // 57089829443294003426484842201820096148621885 43
7           // 13292113578707878773929124160507740 16
8           //
               45231220154454637675796908818182505158069971030869231700931137984020034 3
9           uint256 x =
               105311293498665291426722909308999732236070323463302251608708546560;
10
11          Harness harness = new Harness();
12          uint256 mathMastersOutput = harness.mathMastersTopHalf(x);
13          uint256 solmateOutput = harness.solmateTopHalf(x);
14          console2.log("mathMastersOutput:", mathMastersOutput);
15          console2.log("solmateOutput    :", solmateOutput);
16
17          assert(mathMastersOutput != solmateOutput);
18      }
```

Recommended Mitigation Replace the incorrect value with the correct, fullwidth hexadecimal.

```
1               let r := shl(7, lt
                   (87112285931760246646623899502532662132735, x))
2               r := or(r, shl(6, lt(47223664828696452136 95, shr(r, x))))
3               r := or(r, shl(5, lt(1099511627775, shr(r, x))))
4   -           r := or(r, shl(4, lt(16777002, shr(r, x))))
5   +           r := or(r, shl(4, lt(16777215, shr(r, x))))
```

Low

[L-1] Custom errors don't work in solc 0.8.3

[L-2] Incorrect function selector for `MathMasters::MathMasters__FullMulDivFailed()` custom error used in `MathMasters::mulWad` and `MathMasters::mulWadUp` functions

```
1   -   mstore(0x40, 0xbac65e5b)
2   +   mstore(0x40, 0xa56044f7)
```

Informational

[I-1] Free memory pointer is being overwritten