APEX
SECURITY

Puppy Raffle Audit Report

Version 1.0

Austin Patkos

February 1, 2024

Prepared by: APex Lead Auditors: - Austin Patkos

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…. which may be a hint.

Disclaimer

Austin Patkos 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
	High	H	H/M	M
Likelihood	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.

Findings

Highs

[H-1] Function `MathMasters::mulWadUp` is broken and can return incorrect results.

Summary -

Using increased fuzz testing it was found that the `MathMasters::mulWadUp` function can return incorrect information.

Vulnerability Details

One such case had the inputs as uint256x = 332348412358347524323390 8 and uint256y = 1661742061791737621616955. When used as inputs for `MathMasters::mulWadUp` the result was 552277335985571027172168141646 1. However the correct result is. 5522773359855710271721683078203. Line 56 is noted below. It seems to adjust x under certain conditions, but the logic is not clear. The use of z here is particularly confusing since z is not initialized before this operation.

```
    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, or(div(not(0), y), x))) {
                mstore(0x40, 0xbac65e5b) // `MathMasters__MulWadFailed()`.
                revert(0x1c, 0x04)
            }
@>            if iszero(sub(div(add(z, x), y), 1)) { x := add(x, 1) }
            z := add(iszero(iszero(mod(mul(x, y), WAD))), div(mul(x, y), WAD))
        }
    }
```

Impact

Users of this library can return incorrect information breaking the intented function of the library. Theoretically if a DeFi protocol were to implement this library it could result in funds being lost.

Tools Used

Foundry Fuzz Testing

Recommendations

It is recomended to remove line 56 from the function as described below.

```
    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, or(div(not(0), y), x))) {
                mstore(0x40, 0xbac65e5b) // `MathMasters__MulWadFailed()`.
                revert(0x1c, 0x04)
            }
-           //if iszero(sub(div(add(z, x), y), 1)) { x := add(x, 1) }
            z := add(iszero(iszero(mod(mul(x, y), WAD))), div(mul(x, y), WAD))
        }
    }
```

This new function was ran through an extensive fuzz test of over 100,000,000 inputs and passed each case. It also passed previous input parameters that returned bad information before the function was modified.