Eggstravaganza - Findings Report

Table of contents

Contest Summary

Sponsor: First Flight #37

Dates: Apr 3rd, 2025 - Apr 10th, 2025

See more contest details here

Results Summary

Number of findings:

- High: 1
- Medium: 0
- Low: 0

High Risk Findings

H-01. Weak Randomness Generator Allows Predictable Egg Minting

Summary: The randomness generator implemented in `EggHuntGame::searchForEgg` is weak allowing predictable finding of eggs

Vulnerability Details: The `searchForEgg` function relies on a pseudo-random number generator constructed from block.timestamp, block.prevrandao, msg.sender, and eggCounter, hashed with keccak256 and modulo 100. While this approach generates a seemingly random value, the inputs are predictable rendering the randomness weak and exploitable.

```
function searchForEgg() external {
        require(gameActive, "Game not active");
        require(block.timestamp >= startTime, "Game not started yet");
        require(block.timestamp <= endTime, "Game ended");

     // Pseudo-random number generation (for demonstration purposes only)
        uint256 random = uint256(
        keccak256(abi.encodePacked(block.timestamp, block.prevrandao, msg.sender,
        ) % 100; //@audit -> weak randomness generator

        if (random < eggFindThreshold) {
            eggCounter++;
            eggsFound[msg.sender] += 1;
            eggNFT.mintEgg(msg.sender, eggCounter);
         emit EggFound(msg.sender, eggCounter, eggsFound[msg.sender]);
        }
    }
```

Impact: If the eggNFT tokens have market value (e.g., tradable on secondary markets), an attacker could accumulate a disproportionate number of NFTs through this exploit. This could lead to significant financial gain for the attacker at the expense of the project or other players, especially if the NFTs are rare or tied to future utility.

Tools Used: Manual Review, Aderyn

Recommendations: Implement chainlink's VRF for randomness generation