Rock Paper Scissors - Findings Report

Table of contents

Contest Summary

Sponsor: First Flight #38

Dates: Apr 17th, 2025 - Apr 24th, 2025

See more contest details here

Results Summary

Number of findings:

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

Medium Risk Findings

M-01. The joinGameWithEth function in the RockPaperScissors contract does not check if a player has already joined the game, allowing another person to join and overwrite the previous player as playerB

Description : In the `RockPaperScissors` contract, the `joinGameWithEth` function lacks a proper check to verify whether a player has already joined the game. As a result, multiple users can call this function and become `playerB`, which overwrites the previous `playerB` address.

```
 Solidity
function joinGameWithEth(uint256 _gameId) external payable {
        Game storage game = games[_gameId];
        require(game.state == GameState.Created, "Game not open to join");
        require(game.playerA != msg.sender, "Cannot join your own game");
        require(block.timestamp <= game.joinDeadline, "Join deadline passed");
        require(msg.value == game.bet, "Bet amount must match creator's bet");

        game.playerB = msg.sender;
        emit PlayerJoined(_gameId, msg.sender);
    }
```

Impact :

1) The player who joins as `playerB` first and sends ETH can get their spot overwritten by another player, causing them to loss their funds.

Proof of code :

The following test case demonstrates how a second player can overwrite the `playerB` slot in an existing game. The first player (`joiner1`) successfully joins the game, but a second player (`joiner2`) can call `joinGameWithEth` again and replace `playerB` . This test passed, confirming the bug. Logs confirm that the stored `playerB` after the second call is the address of `joiner2` , not `joiner1` , resulting in the first player losing both their game position and their ETH.

```
 solidity
function testjoinmorethan2personingame() external{

    address creator = address(1);
    vm.deal(creator, 1 ether);


    vm.prank(creator);
    uint256 gameid = game.createGameWithEth{value : 1 ether}(1 , 5 minutes);


    address joiner1 = makeAddr("john");
    address joiner2 = makeAddr("wick");
    vm.deal(joiner1 , 1 ether);
    vm.deal(joiner2 , 1 ether);

    vm.prank(joiner1);
    game.joinGameWithEth{value:1 ether}(gameid);

    console.log("first joiner address" , joiner1);


    vm.prank(joiner2);

    game.joinGameWithEth{value:1 ether}(gameid);

    (address storedPlayerA, address storedPlayerB,,,,,,,,,,,,,, RockPaperScisso

    console.log("second joiner of the game" , storedPlayerB);
    console.log("second joiner of the game" , joiner2);



    }
```

Test Result Explained (PoC Output)

```
 Solidity
[PASS] testjoinmorethan2personingame() (gas: 261562)
Logs:
  first joiner address 0x0b80612770101Db7f47919628857D749FA7dd359
  second joiner of the game 0x7b4A4e774605C2399eb8e9ef15A2E56538638048
  second joiner of the game 0x7b4A4e774605C2399eb8e9ef15A2E56538638048
```

Simple Explanation of What This Proves:

- First, a game is created by `creator`.
- Then, `joiner1` joins the game and becomes `playerB`.
- After that, `joiner2` also joins the **same game**, and **overwrites `playerB`**.
- The `storedPlayerB` (read from the contract) matches the second joiner's address, proving that `joiner1` got replaced.

- This shows that **there’s no check in `joinGameWithEth` to prevent multiple people from joining the same game as `playerB`**, causing the first joiner to lose their position and their ETH.

Tools Used :

1)Vs code

2)Manual review

Recommendations : To prevent multiple players from joining the same game and overwriting the existing `playerB`, add a check to ensure that `playerB` is empty before assigning a new one.

```
Solidity
        function joinGameWithEth(uint256 _gameId) external payable {
        Game storage game = games[_gameId];

        require(game.state == GameState.Created, "Game not open to join");
        require(game.playerA != msg.sender, "Cannot join your own game");
        require(block.timestamp <= game.joinDeadline, "Join deadline passed");
        require(msg.value == game.bet, "Bet amount must match creator's bet");
    ++  require(game.playerB == address(0), "A player has already joined this ga

        game.playerB = msg.sender;
        emit PlayerJoined(_gameId, msg.sender);
    }
```