2024-12-christmas-dinner Audit Report

Version 1.0

Vincent71399

Dec 26, 2024

2024-12-christmas-dinner Audit Report

Vincent71399

Dec 26, 2024

2024-12-christmas-dinner Audit Report

Auditor:

- Vincent71399

Protocol Summary

Platform:

- CodeHawks

Disclaimer

I, Vincent71399, make all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the 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

Findings

High

[H-1] ETH may be locked in the contract, fund may loss

Description: The contract does not have a way to withdraw the funds after the deadline has passed. even the host cannot withdraw the Eth. This means that the funds will be locked in the contract.

Impact: Loss of Funds

Recommended Mitigation: Add a function to let the host withdraw the funds after the deadline has passed.

```
+   event BeforeDeadline();

    ...
+   modifier EventEnded() {
+       if(block.timestamp <= deadline) {
+           revert BeforeDeadline();
+       }
+       _;
+   }
    ...

+   function withdrawETH() external onlyHost EventEnded {
+       require(address(this).balance > 0, "No funds to withdraw");
+       payable(host).transfer(address(this).balance);
+   }
```

[H-2] Host can set deadline multiple times

Description: The `deadlineSet` variable is never set to true, so the host change the deadline at any time.

```
    function setDeadline(uint256 _days) external onlyHost {
        // this will never trigger
@>      if(deadlineSet) {
            revert DeadlineAlreadySet();
        } else {
            deadline = block.timestamp + _days * 1 days;
            emit DeadlineSet(deadline);
        }
    }
```

Impact: The host can extend the deadline, or immediately end the event.

Proof of Concept: 1. HOST can extend the deadline by 10 days. 2. HOST can end the event immediately.

Add the following code into `ChristmasDinner.t.sol`:

```
    function test_hostChangeDeadline() public {
        vm.startPrank(deployer);
        uint256 oldDeadline = cd.deadline();
        // extend deadline by 10 days
        cd.setDeadline(DEADLINE + 10);
        uint256 newDeadline = cd.deadline();
        assertGe(newDeadline, oldDeadline);
        // end the event
        cd.setDeadline(0);
        assertEq(cd.deadline(), block.timestamp);
        vm.stopPrank();
    }
```

Recommended Mitigation: change 'deadlineSet' to true after the deadline is set.

```
    function setDeadline(uint256 _days) external onlyHost {
        // this will never trigger
        if(deadlineSet) {
            revert DeadlineAlreadySet();
        } else {
+           deadlineSet = true;
            deadline = block.timestamp + _days * 1 days;
            emit DeadlineSet(deadline);
        }
    }
```

Medium

[M-1] HOST can withdraw tokens when event is going on, breaking the refund mechanism

Description: host can withdraw tokens when event is going on, cause participants not being able to refund their tokens.

```
    function withdraw() external onlyHost {
        address _host = getHost();
        i_WETH.safeTransfer(_host, i_WETH.balanceOf(address(this)));
        i_WBTC.safeTransfer(_host, i_WBTC.balanceOf(address(this)));
        i_USDC.safeTransfer(_host, i_USDC.balanceOf(address(this)));
    }
    ...
    function refund() external nonReentrant beforeDeadline {
        address payable _to = payable(msg.sender);
        _refundERC20(_to);
        _refundETH(_to);
        emit Refunded(msg.sender);
    }
```

if the contract does not have enough tokens to refund, the refund will be reverted.

Impact: Participants may not be able to refund their tokens.

Proof of Concept: 1. Participant sends some ERC20 token (in the whitelist) to the contract. 2. Host withdraws all the tokens. 3. Participant tries to refund the tokens.

```
    import {IERC20Errors} from "../lib/openzeppelin-contracts/contracts
       /interfaces/draft-IERC6093.sol";
    ...
    function test_hostWithdrawBeforeEventEndBreakParticipantRefund()
       public {
        vm.prank(user1);
        cd.deposit(address(wbtc), 5000);

        vm.prank(deployer);
        cd.withdraw();

        vm.prank(user1);
        vm.expectRevert(abi.encodeWithSelector(
            IERC20Errors.ERC20InsufficientBalance.selector,
            address(cd),
            0, // Current balance
            5000 // Attempted transfer amount
        ));
        cd.refund();
    }
```

Recommended Mitigation: Add a modifier to prevent the host from withdrawing tokens when the event is going on.

```
+   modifier beforeEventEnd() {
+       require(block.timestamp <= deadline, "Event has ended");
+       _;
+   }
    ...
-   function withdraw() external onlyHost {
+   function withdraw() external onlyHost EventEnded {
        address _host = getHost();
        i_WETH.safeTransfer(_host, i_WETH.balanceOf(address(this)));
        i_WBTC.safeTransfer(_host, i_WBTC.balanceOf(address(this)));
        i_USDC.safeTransfer(_host, i_USDC.balanceOf(address(this)));
    }
```