Protocol Audit Report

Version 1.0

Zac Williamson

March 21, 2025

First Flight | Voting Booth

Zac Williamson

March 21, 2025

Prepared by: Zac Williamson Lead Auditor: - Zac Williamson

Table of Contents

Protocol Summary

This is a special rendition of CodeHawks First Flights, as this contract is a simplified version of a real contract which was audited by Cyfrin in a private audit and contained the same bug! Thanks to Dacian for creating this First Flight!

Your mission, should you choose to accept it, is to find that bug!

This contract allows the creator to invite a select group of people to vote on something and provides an eth reward to the for voters if the proposal passes, otherwise refunds the reward to the creator. The creator of the contract is considered "Trusted".

This contract has been intentionally simplified to remove much of the extra complexity in order to help you find the particular bug without other distractions. Please read the comments carefully as they note specific findings that are excluded as the implementation has been purposefully kept simple to help you focus on finding the harder to find and more interesting bug.

This contract intentionally has no time-out period for the voting to complete; lack of a time-out period resulting in voting never completing is not a valid finding as this has been intentionally omitted to simplify the codebase.

Disclaimer

Zac Williamson will make 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
Likelihood	High	H	H/M	M
	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.

Audit Details

The findings described in this document correspond the following commit hash

```
5b9554656d53baa2086ab7c74faf8bdeaf81a8b7
```

Scope

```
./src/
#-- VotingBooth.sol
```

Roles

- `creator` - Deployer of the protocol, they are a trusted used who will receive the funds if a vote fails.
- `AllowedVoters` - A list of addresses that are allowed to vote on proposals.

Issues found

Severity	Number of issues found
High	2
Medium	0
Low	0
Info	0
Gas	0
Total	2

Findings

High

[H-1] `VotingBooth::_distributeRewards` to "for" voters leaves ETH leftover and locked in `VotingBooth` contract.

Description: When a proposal gets passed, the rewards are calculated using the `totalVotes` instead of `totalVotesFor`, not distributing the full reward amount to the `s_votersFor` and leaving leftover funds in the `VotingBooth` contract.

```
1      // distributes rewards to the `for` voters if the proposal has
2      // passed or refunds the rewards back to the creator if the
           proposal
3      // failed
4      function _distributeRewards() private {
5          uint256 totalVotesFor = s_votersFor.length;
6          uint256 totalVotesAgainst = s_votersAgainst.length;
7          uint256 totalVotes = totalVotesFor + totalVotesAgainst;
8          uint256 totalRewards = address(this).balance;
9
10         if (totalVotesAgainst >= totalVotesFor) {
11             _sendEth(s_creator, totalRewards);
12         }
13         else {
14 @>             uint256 rewardPerVoter = totalRewards / totalVotes;
15
16             for (uint256 i; i < totalVotesFor; ++i) {
17                 if (i == totalVotesFor - 1) {
18                     rewardPerVoter = Math.mulDiv(totalRewards, 1,
                           totalVotes, Math.Rounding.Ceil);
19                 }
20                 _sendEth(s_votersFor[i], rewardPerVoter);
21             }
22         }
23     }
```

Impact: Users that voted "for" the proposal don't recieve the full-rewards. The leftover funds are locked in `VotingBooth` because there is also no `withdraw` function.

Proof of Concept:

1. New `booth` is created with 5 users and 10 ETH of rewards
2. 2 users vote for, 1 user votes againt, the `booth` closes and rewards are distributed
3. "For" users are rewarded `3.33` eth and `3.33` ETH is leftover in the contract
4. Leftover ETH can't be taken out of contract without `withdraw` function

Paste test in 'VotingBoothTest.t.sol

Test

```
function testIfPeopleVoteForFundsAreDistributedToEachForVoterAndContractBalanceIsZero() public {
        uint256 startingAmount = address(this).balance;
        console.log("Creator Balance BEFORE: ", startingAmount);
        console.log("- - - - - - - - - - - - - - - - - - - - - - - - -");

        vm.prank(address(0x1));
        booth.vote(true);

        vm.prank(address(0x2));
        booth.vote(true);

        vm.prank(address(0x3));
        booth.vote(false);

        assert(!booth.isActive());

        console.log("(FOR) address(0x1).balance: ", address(0x1).balance);
        console.log("(FOR) address(0x2).balance: ", address(0x2).balance);
        console.log("(AGAINST) address(0x3).balance: ", address(0x3).balance);
        console.log("- - - - - - - - - - - - - - - - - - - - - - - - -");

        console.log("address(booth).balance: ", address(booth).balance);
        console.log("- - - - - - - - - - - - - - - - - - - - - - - - -");

        console.log("Creator Balance AFTER: ", address(this).balance);
}
```

Logs

```
Creator Balance BEFORE:  0
- - - - - - - - - - - - - - - - - - - - - - - - -
(FOR) address(0x1).balance:  3333333333333333333
(FOR) address(0x2).balance:  3333333333333333334
(AGAINST) address(0x3).balance:  0
- - - - - - - - - - - - - - - - - - - - - - - - -
address(booth).balance:  3333333333333333333
- - - - - - - - - - - - - - - - - - - - - - - - -
Creator Balance AFTER:  0
```

Recommended Mitigation:

1. Add a `withdraw` function
2. Calculate rewards using `totalVotesFor` instead of `totalVotes`:
    1. NOTE: only works with this section commented out:

```
if (i == totalVotesFor - 1) {
     rewardPerVoter = Math.mulDiv(totalRewards, 1, totalVotes, Math.
        Rounding.Ceil);
 }
```

Implementation:

```
 function _distributeRewards() private {
     uint256 totalVotesFor = s_votersFor.length;
     uint256 totalVotesAgainst = s_votersAgainst.length;
     uint256 totalVotes = totalVotesFor + totalVotesAgainst;
     uint256 totalRewards = address(this).balance;

     if (totalVotesAgainst >= totalVotesFor) {
         _sendEth(s_creator, totalRewards);
     }
     else {
-         uint256 rewardPerVoter = totalRewards / totalVotes;
+         uint256 rewardPerVoter = totalRewards / totalVotesFor;

         for (uint256 i; i < totalVotesFor; ++i) {
             if (i == totalVotesFor - 1) {
                 rewardPerVoter = Math.mulDiv(totalRewards, 1,
                    totalVotes, Math.Rounding.Ceil);
             }
             _sendEth(s_votersFor[i], rewardPerVoter);
         }
     }
 }
```

Logs

```
Creator Balance BEFORE:  0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(FOR) address(0x1).balance:  5000000000000000000
(FOR) address(0x2).balance:  5000000000000000000
(AGAINST) address(0x3).balance:  0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
address(booth).balance:  0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Creator Balance AFTER:  0
```

[H-1] Contract locks Ether without a withdraw function

It appears that the contract includes a payable function to accept Ether but lacks a corresponding function to withdraw it, which leads to the Ether being locked in the contract. To resolve this issue, please implement a public or external function that allows for the withdrawal of Ether from the contract.

1 Found Instances

- Found in src/VotingBooth.sol Line: 36

```
1  contract VotingBooth {
```