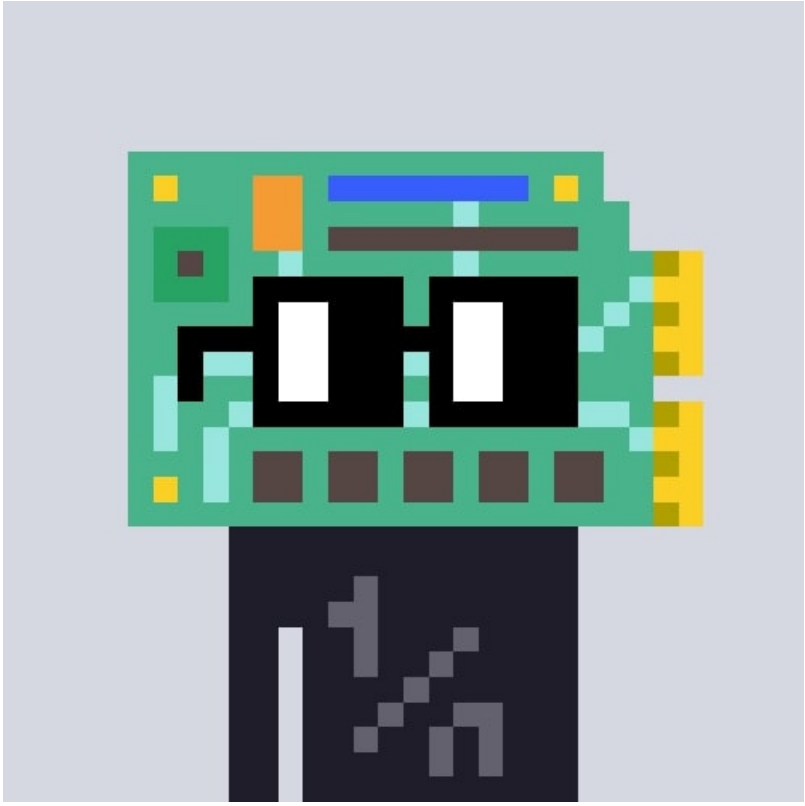

Prepared by: 0xnightfall.eth

Lead Auditors:

- 0xnightfall.eth

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
  - Scope
  - Roles
- Executive Summary
  - Issues found
- Findings
- High
- Medium
- Low
- Informational
- Gas

Protocol Summary

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

The 0xnightfall.eth team makes 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
	High	H	H/M	M
Likelihood	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

- Commit Hash: 5b9554656d53baa2086ab7c74faf8bdeaf81a8b7
- In Scope:

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.

Executive Summary

Issues found

Findings

High

[H-1] Calculation of `VotingPool::rewardPerVoter` is wrong leading to native assets getting stuck in the contract

Description: The calculation of the `VotingPool::rewardPerVoter` is done incorrectly. As per the documentation *the rewards are to be distributed to the **For voters***.

But when calculating the rewardPerVoter in Line: 192, it divides the rewards by totalVotes and not by totalForVotes.

```
uint256 rewardPerVoter = totalRewards / totalVotes;
```

Further down the code in Line: 2 the rewards are distributed to the voters who voted For the proposal.

```
_sendEth(s_votersFor[i], rewardPerVoter);
```

Impact: This results in significant amounts of the rewards being stuck in the contract forever and doesn't go to the users who voted *For* the proposal.

Proof of Concept:

1. Proposal is created with 5 allowed voters
2. Voters 1 and 2 vote *For* the proposal
3. Voter 3 votes *Against* the proposal.
4. Quorum is reached and rewards are calculated incorrectly
5. Instead of Voters 1 and 2 getting half of the rewards each, they only get 1/3rd each. While the other 1/3rd is stuck in the contract forever.

▶ **Proof of Code**

```
function testRewardsGetStuck() external {
        deal(address(this), ETH_REWARD);
        booth = new VotingBooth{value: ETH_REWARD}(voters);
        vm.prank(address(0x1));
        booth.vote(true);
```

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

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

        assert(address(booth).balance > 0);
    }
```

Recommended Mitigation:

```
- uint256 rewardPerVoter = totalRewards / totalVotes;
+ uint256 rewardPerVoter = totalRewards / totalForVotes;
```

Medium

[M-1] Solidity Version 0.8.20 and above is not supported by Arbitrum

Description: Solc compiler version 0.8.20 and above switches the default target EVM version to Shanghai, which means that the generated bytecode will include PUSH0 opcodes. Since the project wants to deploy to Arbitrum, but since Arbitrum doesn't have PUSH0 opcode support (as per Arbitrum documentation) deployment of the contract will fail.

- Found in src/VotingBooth.sol Line: 2

```
pragma solidity ^0.8.23;
```

Impact: The project will not be able to deploy on Arbitrum as it doesn't support PUSH0 opcode.

Recommended Mitigation: The issue can be mitigated by changing the solidity version to `0.8.19`.

Informational

[I-1] `VotingBooth::DISALLOWED` variable never used

Description: The variable `VotingBooth::DISALLOWED` is never used anywhere in the contract. Hence it can removed.

Recommended Mitigation:

```
-     uint8 private constant DISALLOWED = 0;
```

Gas

[G-1] `VotingBooth::s_creator` variable can be marked as immutable

Description: The `VotingBooth::s_creator` variable can be marked as immutable as it is not updated after initialization in constructor. Hence this will reduce the gas cost of reading the variable.

Impact: Gas costs to read the variable will be reduced if changing the variable `s_creator` into an immutable variables.

Recommended Mitigation:

```
- address private s_creator;
+ address private immutable i_creator;
```

[G-2] `VotingBooth::s_totalAllowedVoters` variable can be marked as immutable

Description: The `VotingBooth::s_totalAllowedVoters` variable can be marked as immutable as it is not updated after initialization in constructor. Hence this will reduce the gas cost of reading the variable.

Impact: Gas costs to read the variable will be reduced if changing the variable `s_totalAllowedVoters` into an immutable variables.

Recommended Mitigation:

```
- address private s_totalAllowedVoters;
+ address private immutable i_totalAllowedVoters;
```