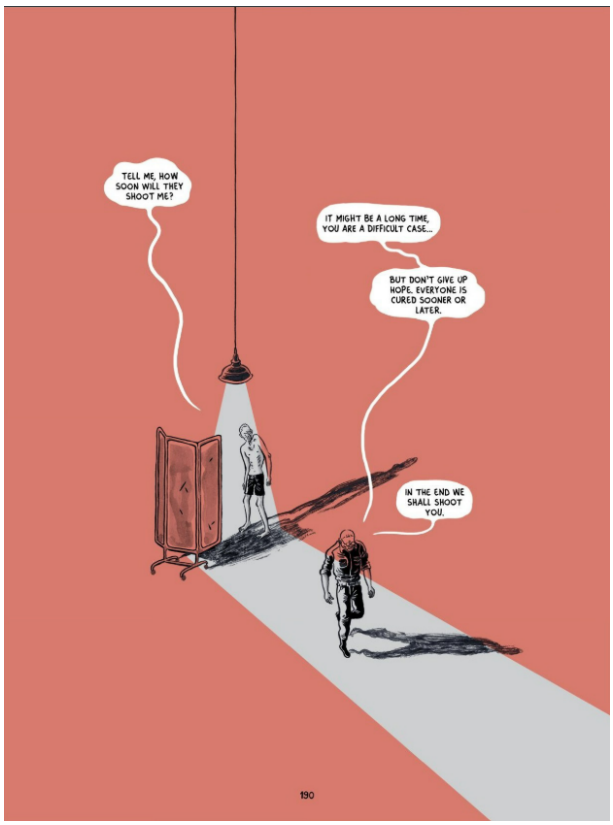

Protocol Audit Report

Version 1.0

Cyfrin.io

November 30, 2024

Protocol Audit Report

idir

November 29, 2024

Prepared by: idir badache Lead Auditors: idir badache (some day in the future) - xxxxxxx

Protocol Summary

GivingThanks is a decentralized platform that embodies the spirit of Thanksgiving by enabling donors to contribute Ether to registered and verified charitable causes. Charities can register themselves, and upon verification by the trusted admin, they become eligible to receive donations from generous participants. When donors make a donation, they receive a unique NFT as a donation receipt, commemorating their contribution. The NFT's metadata includes the donor's address, the date of the donation, and the amount donated.

Disclaimer

The IDIR dream 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
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.

Issues found

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

Findings

High

[#-H] A wrong implementation of the constructor in the `GivingThanks` contract.

Description: the unused local variable in the construcotr definition `_registry` is meant to be the address of the `CharityRegistry` contract instance. In this constructo the `CharityRegistry` instance address would be the msg.sender, meaning the dployer of the `GivingThanks` contract.

Impact: The `GivingThanks` contract has a complete depandency on `CharityRegistry`, this error could lead to corrupted protocol functionality.

Proof of concept: If you go to the `GivingThanksTest` you'll find:

```
vm.prank(admin);
    charityContract = new GivingThanks(address(registryContract));
```

This will not assign the state variable charityContract to a valid registryContract instance; instead, it assigns it to whoever the deployer is (i.e., the caller of the constructor).

Recommended Mitigation: in the `GivingThanks` contract construcotr change the implementation to uset the `_registry` local variable. the code :

```
    constructor(address _registry) ERC721("DonationReceipt", "DRC") {
        registry = CharityRegistry(_registry);// Must be a valid
           address of a CharityRegistry contract.
        owner = msg.sender;
        tokenCounter = 0;
    }
```

[#-H] NO Access control for updating the CharityRegistry

Description: In `GivingThanks::updateRegistry` this function can be called by anyone.

Impact: It can cause protocol manipulation through malicious, unauthorized changes.

Recommended Mitigation: it depends on the protocol policy on who has the authority to change the CharityRegistry contract for example:

```
    modifier allowedEntity() {
        if (msg.sender != /*adminforExample*/) { //Here, you can
           specify the address of the entity authorized to make changes
            revert GvingThanks__UnauthorizedChange(); //use custom
               errors to save on gas
        }
```

```
5           _;
6       }
7
8       function updateRegistry(address _registry) public allowedEntity{ //
            implement the modifier
9           registry = CharityRegistry(_registry);
10      }
```

Low (informationl)

[#-informational] lacking Documentation and explicit function description

Impact: The protocol might be implicit to other users, which could disrupt teamwork flow. to `avoid` ambiguity add natspec before functions