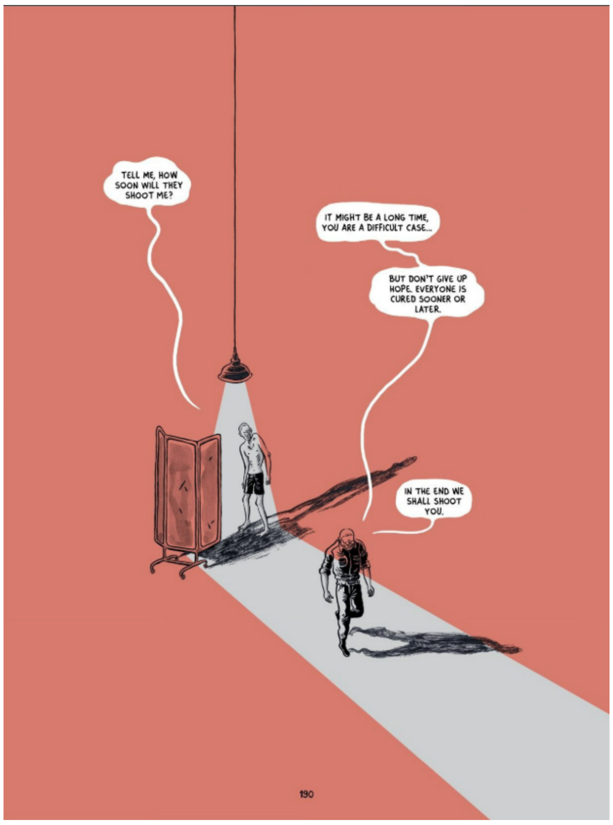
TELL ME, HOW SOON WILL THEY SHOOT ME?
IT MIGHT BE A LONG TIME, YOU ARE A DIFFICULT CASE...
BUT DON'T GIVE UP HOPE. EVERYONE IS CURED SOONER OR LATER.
IN THE END WE SHALL SHOOT YOU.
190

# Protocol Audit Report

Version 1.0

*Cyfrin.io*

November 29, 2024

# Protocol Audit Report

idir

November 29, 2024

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

## Table of Contents

## Protocol Summary

this protocol claims that it allow users to create and store there passwords completely onchain. creators of password have complete and only ownability and transparency.

## 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 |
| | 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:

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

### Scope

- Commit Hash: 2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
- In Scope:

```
./src/
#-- PasswordStore.sol
```

- Solc Version: 0.8.18
- Chain(s) to deploy contract to: Ethereum

### Roles

- Owner: The user who can set the password and read the password.
- Outsider: No one else should be able to set or read the password. # Executive Summary

### Issues found

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

## Findings

### High

#### [H-#] Storing the password on-chain makes it visible to anyone

**Description:** All data stored on-chain is visible to anyone, and I can be read directly from the blockchain. the `PasswordStore::s_password` variable is intended to be a private variable and only accessed through the `PasswordStore::getPassword` function wish only the owner is authorized to call it.

We show one such method of reading any data off chain below. **Impact:** Anyone can read the private password, severely breaking the functionality of the protocol.

**Proof Of Concept:**

The below test case shows how anyone can effectively read a password that meant to be private, from the block chain.

1. start a locally running chain

```
make anvil
```

2. Deploy the contract on the local chain

```
make deploy
```

3. using cast interact with the contracts storage by running

**Note:** make sure to locate the storage slot of wish data is supposed to be private.

```
cast storage <contract-address> <storage-slot> <network-args>
```

this would return bytes32 hash of the storage slot data.

4. Using cast to convert bytes32 to string

```
cast parse-bytes32-string <bytes32-data>
```

### [H-#] `PasswordStore::setPassword` no access control. unauthroized entities can setPassword.

**Description:** The `PasswordStore::setPassword` cannot prevent non owners to set a new password. making the password aka 'private data' accessible to everyone. thus an ownablity checking must be implemented.

```
    function setPassword(string memory newPassword) external {
        s_password = newPassword;
        emit SetNetPassword();
    }
```

**Proof of Concept:** you'll notice a test added (more like an attack scenario) that proofs if a nonOwner can set a password, you may check it out.

**Recommended Mitigation:** adding access control implementation to the `PasswordStore::setPassword` simply check wether the `msg.sender` is the actual owner or for reusability and clarity add a modifier like so:

```
modifier onlyOwner() {
    if (msg.sender != s_owner) {
        revert PasswordStore_notOwner();
    }
    _;
}
function setPassword(string memory newPassword) external onlyOwner{
    s_password = newPassword;
    emit SetNetPassword();
}
```