\begin{titlepage} \centering \begin{figure}[h] \centering \includegraphics[width=0.5\textwidth]{logo.pdf} \end{figure} \vspace*{2cm} {\Huge\bfseries Protocol Audit Report\par} \vspace{1cm} {\Large Version 1.0\par} \vspace{2cm} {\Large\itshape hossein.io\par} \vfill {\large \today\par} \end{titlepage}

\maketitle

Prepared by: Hossein Ahmadipoor Lead Auditors:

- Hossein Ahmadipoor

Table of Contents

Protocol Summary

PasswordStore is a protocol dedicated to storage and retrieval of a user's password. The protocol is designed to be used by a single user, and is not designed to be used by multiple users. Only the owner should be able to set and access this password.

Disclaimer

The YOUR_NAME_HERE 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

The findings described in this document correspond the following commit hash;

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

Scope

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

Roles

Owner: The user who can set the password and read the password. Outsides: No one else should be able to set or read the password.

Executive Summary

I spend 24 hours reviewing the security issues; I've found 2 high sevierity volunerabilities and one low.

Issues found

Sevierity	Number of issues found
High	2
Medium	0
Low	0
Info	1
Total	3

Findings

High

[H-1] Storing password on-chain makes it visible to anyone, and no longer private

Description: All data stored on-chain is visible to anyone, and 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, whih is intended to be only called by the owner of the contract. We show one such method of reading any data off chain below.

Impact: Anyone can read the private password, severly breaking the functionality of the protocol.

Proof of Concept: (Proof of Code) The below test case shows how anyone can read the password directly from the blockchain.

1. Create a locally running chain

```
make anvil
```

2. Deploy the contract to the chain

```
make deploy
```

3. Run the storage tool We use `1` because that's the storage slot of the `s_password` in the contract.

```
cast storage <ADDRESS_HERE> 1 --rpc-url http://127.0.0.1:8545
```

You'll get an output that looks like this:

`0x6d7950617373776f726400000000000000000000000000000000000000000014`

You can then parse that hex to a string with the following command

```
cast parse-bytes32-string
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

and get output of `myPassword`

Recommended Mitigation: Due to this, the overall architecture of the contract should be rethought. One could encrypt the password off-chain and then store the encrypted password on-chain. Howerver, you'd also likely want to remove the view function as you wouldn't want the user to accidently send transaction with the password that decrypts your password.

[H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the password

Description: The `PasswordStore::setPassword` function is set to be an `external` function, however, the natspec of the function and overall purpose of the smart contract is that `This function allows only the owner to set a new password.`

```
    function setPassword(string memory newPassword) external {
@>      // @audit - There are now access controls
        s_password=newPassword;
        emit SetNetPassword();
    }
```

Impact: Anyone can set/change the password of the contract, severly breaking the contract intended functionality.

Proof of Concept: Add the following to the `passwordStore.t.sol` test file

▶ Code

```
function test_anyone_can_set_password(address randomAddress) public {

        vm.prank(randomAddress);
        string memory expectedPassword = "myNewPassword";
        passwordStore.setPassword(expectedPassword);
        vm.prank(owner);
        string memory actualPassword=passwordStore.getPassword();
        assertEq(actualPassword, expectedPassword);
}
```

Recommended Mitigation: Add an access controll conditional to the setPassword function.

```
    if (msg.sender!=s_owner){
        revert PasswordStore_NotOwner();
    }
```

Informational

[I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect

Description:

```
    /*
     * @notice This allows only the owner to retrieve the password.
     * @param newPassword The new password to set.
     */
    function getPassword() external view returns (string memory) {
```

The `PasswordStore::getPassword` function signature is `getPassword()` which the natspec say it should be `getPassword(string)`

Impact: The natspec is incorrect.

Recommended Mitigation: Remove the incorrect natspec

```diff
-    * @param newPassword The new password to set.
```

Gas