Prepared by: [Rubén Cruz]

Table of Contents

Protocol Summary

PasswordStore is a simple contract dedicated to storage and retrieval of a user´s password. It allows only the owner to change it.

Disclaimer

Rubén Cruz 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

Severity	Number of issues found
High	2
Medium	0
Low	0
Info	1

Severity	Number of issues found
Total	3

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

The findings described correspond to the following commit hash:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

```
./src/PasswordStore.sol
```

Roles

-Owner -Outsiders

Findings

High

[H-1] Variables stored in storage-onchain are visible to everyone. The password is not really private.

Description: All data stored on-chain is visible to anyne, and can be read directly form the blockchain. `PasswordStore::s_password` is intended to be private and only accessed through the `PasswordStore::getPassword` function, which is intended to be called only by the owner. We show one such method of reading any data on-chain below.

Impact: Anyone can read the "private" password.

Proof of Concept: This test below shows how anyone can read the password:

1. Create a locally running chain

```
make anvil
```

1. Deploy the contract

```
make deploy
```

1. Run the storage tool.

we use `1` because thet is the storage slot for s_password.

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

You will get an output that looks like this:
`0x6d7950617373776f726400000000000000000000000000000000000000000014`

Then, you can parse that hex to string with:

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

You will get `mypassword`.

Recommended Mitigation: Due to this, the overall architecture of the contract should be rethought. One could encrypt the password off-chain and store it on-chain, and only decrypt it when needed (with the requirement of another password off-chain to decrypt the password). However, you would also likely want to remove the view function, as you would not want the user to accidentally send a tx with the password that decrypts your password.

[H-2] `PasswordStore::setPassword` has no access control, allowing a non-owner to change the password.

Description: The function `PasswordStore::setPassword` does not check that the owner is the one setting the password, allowing a non-owner to change the password and breaking the protocol logic.

```
    function setPassword(string memory newPassword) external {
      @> // @audit-issue: no access control.
          s_password = newPassword;
          emit SetNetPassword();
      }
```

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

Proof of Concept: Add the following to the `PasswordStore.t.sol` contract:

▶ Details

```

```

```
    function test_anyone_can_set_password(address randomAddress) public {
        vm.assume(randomAddress != owner);
        vm.prank(randomAddress);
        string memory expectedPassword = "myNewPassword";
        passwordStore.setPassword(expectedPassword);

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

Recommended Mitigation: Add access control conditional to the `PasswordStore::setPassword` function. You can add this code:

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

Informational

[I-1] The `PasswordStore::getPassword` function natspec indicates a parameter that does not exist, making it incorrect.

Description:

```
    /*
    * @notice This allows only the owner to retrieve the password.
@>  * @param newPassword The new password to set.
    */

    // @audit-issue: There is no parameter newPassword

    function getPassword() external view returns (string memory) {
```

The `PasswordStore::getPassword` function signature is `getPassword()`. It should be `getPassword(string)`.

Impact: Incorrect NATSpec.

Recommended Mitigation: Remove the natspec line:

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

or:

add the parameter:

```
+    function getPassword() external view returns (string memory) {
```