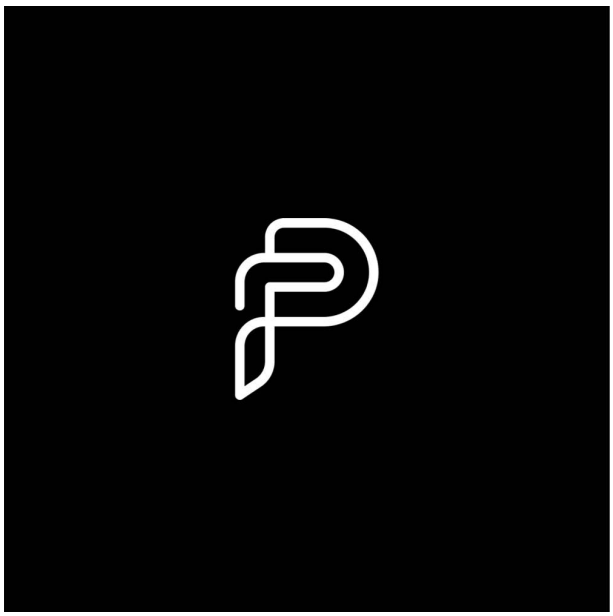

Protocol Audit Report

Version 1.0

Plairfx

February 28, 2024

PasswordStore Audit Report

Plairfx

27 Feb 2024

Prepared by: [PlairFx](twitter.com/PlairFx Lead Auditors: - PlairFx

Table of Contents

Protocol Summary

PasswordStore is a protocol that lets users store their passwords and retrieve it while is unaccessible to other users.

Disclaimer

PlairFx 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:

```
2e8f81e
```

Scope

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

Roles

Owner - Can retrieve the password and store password Others - No one can read or set the Owners passsword.

Executive Summary

Issues found

Serverity	Number Of Issues found
Highs	2
Mediums	0
Low	0
Info	1
Total	3

Findings

High

[H-01] Storing the password on-chain makes it visible to anyone, and is not private.

Description: All data in stored on-chain is visible to everyone and can be read by everyone from the blockchain, the `PasswordStore: s_password` is intended to be a private variable and only to be accessed by the `PasswordStore: getPassword` function, which should only be viewed by the owner.

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. Deploy Local Chain

Run a local running node with command `anvil`

2. Deploy the contract to the anvil chain

```
make deploy
```

3. run the storage tool

```
cast storage `contract-address` (the storage number) 1 --rpc-url http://127.0.0.1:8545
```

4. Cast Tool take the bytes number and use cast parse-bytes32-string `bytesnumber` and you will the password -> `MyPassword` made in the `DeployPasswordStore.s.sol` file.

This shows the password

Recommended Mitigation: Due to this, overall design of this protocol storing passwords on-chain is impossible without using any encryption, You could try to have the encrypted password on-chain but this would cause to rethink the whole structure needed to be able to do this.

I believe you need to design a new plan that makes it viable to store their password on-chain as this a blockchain.

[H-02] `PasswordStore:setPassword` has no access controls, meaning a non owner can change the password. (Root Cause + Impact)

Description: `PasswordStore:setPassword` is open to everyone because if missing any access controls so only the owner could withdraw, this means everyone can set a password for someone meaning changing the password for someone else.

```
function setPassword(string memory newPassword) external {
->   //@audit This is open to everyone, no control of who has access and not.
        s_password = newPassword;
        emit SetNetPassword();
    }
```

Impact: Anyone can set and change the password of another, essentialy breaking the use case of the protocol.

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

Code

```
    function test_anyone_can_setpassword(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 an access control conditional to the `setPassword` function.

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

Informational

[I-01] TITLE (Root Cause + Impact) The `PasswordStore:getPassword` natspec indicates a parameter is present that does not exist in the file, causing it to be wrong.

Description: `PassordStore:getPassword` natspec indicates a parameter.

```
    /*
     * @notice This allows only the owner to retrieve the password.
    -> * @param newPassword The new password to set.
     */
    function getPassword() external view returns (string memory) {
        if (msg.sender != s_owner) {
            revert PasswordStore__NotOwner();
        }
        return s_password;
    }
```

The`PassordStore:getPassword` function signature is `getPassword()` which the natspec says it should be `getPassword(string)`.

Impact: Wrong Documentation.

Recommended Mitigation: Remove the incorrect natspec line

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