1. PasswordStore Audit

> https://github.com/Cyfrin/security-and-auditing-full-course-s23?tab=readme-ov-file#%EF%B8%8F-section-3-your-first-audit-security-review--passwordstore-audit

[H-1] Password data is stored on-chain and is therefore publicly visible and not private

---

Description
The variable `PasswordStore::s_password` variable uses the `private` modifier to encapsulate the variable within the current contract and restrict programatic access to the variable, however this does not make the actual password value private as all data stored on-chain is publicly visible.

Impact The password is stored in plain text on-chain and can be read by anyone breaking a key tenet of the protocol as described in the Protocol NatSpec.

```
* @notice This contract allows you to store a private password that others won't be able to see.
```

Proof of Concept
We can read the data stored on-chain using the following `foundry` commands:

```
1. Use the foundry anvil to run a local chain and set the stored password.
```

The code from the deployment script sets the password to `myPassword`.

```
contract DeployPasswordStore is Script {
    function run() public returns (PasswordStore) {
        vm.startBroadcast();
        PasswordStore passwordStore = new PasswordStore();
        passwordStore.setPassword("myPassword");
        vm.stopBroadcast();
        return passwordStore;
    }
}
```

```
3. Identify that the storage for password is at slot 1.
4. Use the foundry cast command to read the storage variable value.
5. Convert the byte data to it's string representation.

cast storage 0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0 1 --rpc-url
http://127.0.0.1:8545
0x6d7950617373776f726400000000000000000000000000000000000000000014

cast parse-bytes32-string 0x6d795
0617373776f726400000000000000000000000000000000000000000014
myPassword
```

Recommended mitigation
If this must be stored on-chain then the value should be strongly encrypted and the encryption key must be stored off-chain. The protocol architecture should be reviewed to determine if a more suitable option is viable.

[H-2] Broken Access control allows anyone to set the password

---

Description
The setPassword function is intended to only allow the owner to set the password, however a missing access control allows any user to call setPassword. This breaks a key tenet of the protocol.

```
    /*
     * @notice This function allows only the owner to set a new password.
     * @param newPassword The new password to set.
     */
    function setPassword(string memory newPassword) external {
@>  // The missing access control check should be here.
    // if (msg.sender != s_owner) {
    //     revert PasswordStore__NotOwner();
    //     return
    // }
        s_password = newPassword;
        emit SetNetPassword();
    }
```

Impact

Anyone can set the password on the contract breaking the key tenet of the protocol.

Proof of Concepts

The following test demonstrates that any user can set the password

```
    /** This test demonstrates the protocol is broken by
    - having a non owner set a password and then,
    - the owner getting that newly set password
    */
    function test_non_owner_can_set_password() public {
        //not owner can set the password (incorrectly implemented)
        vm.startPrank(address(1));
        string memory expectedPassword = "Oh!No!";
        passwordStore.setPassword(expectedPassword);

        //only the owner is allowed to get the password (which is correctly
implemented)
        vm.startPrank(owner);
        string memory actualPassword = passwordStore.getPassword();
        assertEq(actualPassword, expectedPassword);
    }
```

Recommended mitigation

Implement the missing access control in the `PasswordStore::setPassword` function.
