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

Description: All data stored on chain is public and visible to anyone. The `PasswordStore::s_password` variable is intended to be hidden and only accessible by the owner through the `PasswordStore::getPassword` function. I show one such method of reading any data off chain below. :br :br

Impact: Anyone is able to read the private password, severely breaking the functionality of the protocol. :br :br **Proof of Concept:**The below test case shows how anyone could read the password directly from the blockchain. We use foundry's cast tool to read directly from the storage of the contract, without being the owner.

Create a locally running chain

make anvil Deploy the contract to the chain

make deploy Run the storage tool

We use 1 because that's the storage slot of 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:

```
cast parse-bytes32-string 0x6d7950617373776f726400000000000000000000000000000000000000000014
And get an output of:

myPassword
```

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

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

Understand with the help of code

```
function setPassword(string memory newPassword) external {
// @Audit - There are no Access Controls.
s_password = newPassword;
emit SetNewPassword();
}
```

Impact:Anyone can set/change the stored password, severely breaking the contract's intended functionality **Proof of Concept:** Add the following to the PasswordStore.t.sol test file:

Run this test to understand this bug more deeply

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

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

Recommended Mitigation:: Add access control to the function.

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

[I-1] `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()` while the natspec says it should be `getPassword(string)`.

Impact: The netspec is incorrect.

Recommended Mintigation: Remove the incorrect netspec line.

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