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

Description: All date 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 varaible and only accessed through `PasswordStore::getPassword` function, which is intented 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 from the password directly from the blockchain.

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. This would require the user to remember another password off-chain to decrypt the password. However, you'd also likely want to remove the view function as you wouldn't want the user to accidentally send a transaction wiht the password that decrypts your password

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

Description: `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 no access controls
        s_password = newPassword;
        emit SetNewPassword();
    }
```

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

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

▶ Code

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

        vm.prank(owner);
```

```
        assertEq(passwordStore.getPassword(), expectedPassword);

    }
```

Recommended Mitigation: Add an access control conditon to the `setPassword` function.

▶ Code

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

    }
```

[H-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing 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 natspec is incorrect.

Recommended Mitigation:

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