[H-1] Storing the password on-chain makes it public, thus visible to anyone.

Description: All data stored on the blockchain is public and can be read by anyone. The 'PasswordStore::s_password' variable is intended to be private, and accessed only by the c ontract owner through the 'PasswordStore::getPassword()' function. However, the s_password variable is not actually private, and can be read by anyone who has access to the contract' s bytecode. This means that anyone can read the password, which is a security vulnerability .

We show one such method of reading the password in the PoC below.

Impact: Anyone can read the password, which is a security vulnerability that could seve rely compromise the security of the contract.

Proof of Concept: (Proof of Code)

The below code snippet shows how to read the password from the contract.
1. Create a locally running chain using Foundry.
```bash
make anvil
```
2. Deploy the contract.
```bash
make deploy
```
3. Run the storage tool to read the password from the contract.

We use the '1' because the 's_password' variable is at storage slot '1'.

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

```

You'll get an output similar to the below:

'0x6d7950617373776f726400000000000000000000000000000000000000000014'

You can decode the hex string to get the password as follows:

```bash

cast parse-bytes32-string 0x6d7950617373776f726400000000000000000000000000000000000000000014

```
Output:

```
myPassword

```
Recommended Mitigation:
The password should not be stored on-chain. Instead, it should be stored off-chain and only retrieved when needed. This can be done by storing the password in a separate contract or using a secure storage solution.

Likelihood & Impact:
- Likelihood: HIGH.
- Impact: HIGH.
- Severity: HIGH.

[H-2] 'PasswordStore::setPassword' has no access controls, meaning anyone can set the p assword.

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

<details>

<summary>Code</summary>

```javascript
function setPassword(string memory newPassword) external {
    //@audit - There are no access controls on this function.
        s_password = newPassword;
        emit SetNetPassword();
    }
```

</details>

Impact: Anyone can set the password, which is a security vulnerability that could severely compromise the security of the contract.

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

<details>
<summary>Code</summary>

```javascript
    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);
    }
```

</details>
</br>
Recommended Mitigation:
The `setPassword` function should be made internal and only callable by the contract owner. This can be done by changing the visibility of the function to `internal` and adding an `onlyOwner` modifier to the function.

<details>
<summary>Code</summary>

```javascript
function setPassword(string memory newPassword) internal onlyOwner {
    s_password = newPassword;
    emit SetNetPassword();
}
```

</details>

Alternatively, add an access control modifier to the `setPassword` function to ensure that only the contract owner can call it.

<details>
<summary>Code</summary>

```javascript
```

```
if (msg.sender != owner) {
    revert PasswordStore_NotOwner();
}
```
</details>

Likelihood & Impact:
- Likelihood: HIGH.
- Impact: HIGH.
- Severity: HIGH.

Description:
The `PasswordStore::getPassword` natspec indicates a parameter that does not exist, causing the natspec to be inaccurate.
<details>
<summary>Code</summary>

```javascript
/**
 * @notice Get the stored password.
 * @return The stored password.
 */
 function getPassword() external view returns (string memory) {
    return s_password;
```
</details>

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

Impact:
The natspec is inaccurate, which could lead to confusion and misunderstanding of the function's purpose and parameters.

Recommended Mitigation:
The natspec should be updated to reflect the correct function signature. Remove the incorrect natspec parameter.

```diff
-  * @param newPassword The new password to set.
+  * @return The stored password.
```

Likelihood & Impact:
- Likelihood: NONE.
- Impact: HIGH.
- Severity: INFORMATION/GAS/NON-CRITICAL.