Title: PasswordStore Protocol Security Review

Author: Muhammad Faran

Date: June 10, 2025

[H-1] `PasswordStore:s_password` Is Not Truly Private — It Can Be Decoded by Anyone

Description:
While the `s_password` variable in the `PasswordStore` contract is marked `private`, this only restricts access from other contracts. In reality, **all on-chain data is publicly accessible** — including private variables. Anyone with access to the blockchain can decode and read this password using the correct storage slot.

Impact:
Anyone can read the stored password, completely undermining the confidentiality the protocol intends to provide.

Proof of Concept:

1. Read the storage slot of `s_password` (slot 1):

   ```
   $ cast storage 0x5FbDB2315678afecb367f032d93F642f64180aa3 1 --rpc-url
   http://127.0.0.1:8545
   0x6d7950617373776f726400000000000000000000000000000000000000000014
   ```

2. Decode the bytes32 result:

   ```
   $ cast parse-bytes32-string
   0x6d7950617373776f726400000000000000000000000000000000000000000014
   myPassword
   ```

Recommended Mitigation: The architecture should be rethought. Sensitive data like passwords should **never be stored in plaintext on-chain**. Consider off-chain storage with on-chain access controls or encryption techniques.

[H-2] `PasswordStore:setPassword` Function Is Missing Access Control

Description: The `setPassword` function in `PasswordStore` lacks access control, allowing **anyone** to call it and overwrite the password. This breaks the assumption that only the contract owner should set the password.

```
function setPassword(string memory newPassword) external {
    // No access control here
    s_password = newPassword;
    emit SetNetPassword();
}
```

Impact: Any external address can set a new password, effectively hijacking the protocol’s intended usage.

Proof of Concept:

```
function test_anyone_can_set_password() public {
    address attacker = address(0xBEEF);
    vm.prank(attacker);
    passwordStore.setPassword("hackedPassword");
    // The test passes — proving unauthorized users can set passwords.
}
```

Recommended Mitigation: Add an ownership check to restrict this function:

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

[I-1] Incorrect `@param` Tag on `getPassword` Function

Description: The `getPassword()` function includes a `@param` tag for `newPassword`, which is incorrect as the function takes **no parameters**.

Impact: Inaccurate NatSpec documentation may mislead developers and auditors.

Proof of Concept: The current NatSpec comment:

```
/// @param newPassword The new password to set.
function getPassword() public view returns (string memory) {
    ...
}
```

Recommended Mitigation: Remove the incorrect `@param` tag:

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