2025

PROTOCOL AUDIT REPORT

@Siddev09

PasswordStore Contract

[H-1] Storing the `Password` on chain makes it `Visable to Anyone` , and no longer private

Description : all data 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 variable and only accessed through `PasswordStore ::getPassword` function which is intended to be only called by the owner of the contract.

Impact : Anyone can read private Password , severely breaking the functionality of the protocol.

Proof of Concept: (Proof of Code) :

1. First, run your local blockchain using Anvil.

```
anvil
```

2. Next, deploy the smart contract on the local blockchain and copy the deployment address.

3. Then, find the storage of the variable using
`cast storage`.

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

```
`0x6d7950617373776f726400000000000000000000000000000000000000000014`
```

4. Finally, parse the storage value into a string.

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

this gives you the password :

```
myPassword
```

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 encrypt 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 with the password that decrypts your password.

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

Description : The `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 - no access controls
      s_password = newPassword;
      emit SetNetPassword();
 }
```

Impact : Anyone can set/change the password of the contract severalty breaking the contract

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

```
 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);
   }
```

Recommended Mitigation : added access control condition to the `setPassword` function

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

[I-1] The `passwordStore::getPassword` natspec indicates a parameter that doesn't exist causing the natspec to be incorrect

Description :

```
function getPassword()external view returns(string memory){}
```

The `passwordStore::getPassword` function signature is `getPassword` which the natspec say it should be `getPassword(string)`

Impact : the natspec is incorrect

Recommended Mitigation : remove the incorrect natspec line