Protocol Audit Report

Version 1.0

MAYANK.io

November 23, 2025

Protocol Audit Report

Mayank

November 23, 2025

Prepared by: MAYANK MOKTA Lead Auditors:

- Mayank Mokta

Table of Contents

Protocol Summary

PasswordStore is a Protocol which stores one user's password and he can change the password and can view the password. Only the owner shoule be able to change or view the password.

Disclaimer

The MAYANK.io team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

Below we have our commit hash

`63541e54586cca26e0a69b3ddc6b8fed150e7d2d`

Scope

`./src/PasswordStore.sol`

Roles

- Owner: The user who can set password and view the password.
- Outsiders: No one should be able to read and set password expect Owner.

Executive Summary

Add some notes here telling how the audit went

We spent X hours with Y auditors using Z tools. etc

Issues found

Severity	Number of issues found
HIGH	2
MEDIUM	0
LOW	0
INFO	1
TOTAL	3

Findings

High

[H-#] TITLE (Root Cause + Impact) The password in the contract is actually not private on-chain

Description: All the data stored on-chain is actually visible to anyone, in out contract `PasswordStore::s_password` is actually privated but on-chain anyone can access it but in our function `PasswordStore::getPassword` we only want this to be accesed by the owner but as i told you on-chain anyone can access it.

Impact: Anyone from outside can read the password, hence `PasswordStore::s_password` is not actually private hence breaking the functionality of the contract.

Proof of Concept: (Proof of Code):

The next shows how your `PasswordStore::s_password` can be accesed by anyone.

1. Create a locally running chain

```
anvil
```

2. Deploy on local chain

```
make deploy
```

3.

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

Recommended Mitigation: The best way in my opinion to store passwords is not in strings because the are still visible on-chain, you should make the variable into bytes32 so it will be hashed from outside and hence no one can see the real string as of password.

[S-#] TITLE (Root Cause + Impact) There is no access control in function `PasswordStore::setPassword`.

Description: The function `PasswordStore::setPassword` does not have any access control, therefore anyone from outside can call the function and set a new password but we want that onlt the owner should be able to call this function. So it just disturbs the logic of the contract.

Impact: This will create a huge impact as anyone form outside can change the function so its not secured at all.

Proof of Concept: Add the below test to your file `PasswordStoreTest.t.sol` so that which proves anyone can call the function `PasswordStore::setPassword` and set the password.

```
function testAnyoneCanChangePassword(address randomuser) external {
        vm.assume(randomuser != owner);
        vm.prank(randomuser);
        string memory expectedPass = "mayank123";
        passwordStore.setPassword(expectedPass);
        vm.prank(owner);
        string memory pass = passwordStore.getPassword();
        assetEq(pass,expectedPass);
    }
```

Recommended Mitigation: You should surely add access control in your function `PasswordStore::setPassword` so only the owner can call the function and set password, the following lines should be added in your function.

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

Informational

[I-#] TITLE (Root Cause + Impact) The discription written for `PasswordStore::getPassword` is incorrect.

Description: The netspac for the function `PasswordStore::getPassword` says this function shoulh have a parameter but in reality it does not have any parameter added my you.

Impact: The function `PasswordStore::getPassword` does not have a parameter.

Proof of Concept: Below it is shown which netspac is incorrect.

```
/*
     * @notice This allows only the owner to retrieve the password.
@==>     * @param newPassword The new password to set.
     */
```

Recommended Mitigation: Below it is shown which line should be removed.

```

/*
     * @notice This allows only the owner to retrieve the password.
-    * @param newPassword The new password to set.
     */
```