Protocol Audit Report

Version 1.0

Kamil

October 23, 2025

Protocol Audit Report

Kamil Nissar

October 22, 2025

Prepared by: Kamil Lead Security Researcher : - Kamil

Table of Contents

Protocol Summary

PasswordStore is a protocol dedicated to storage and retrieval of a users passwords. The protocol is designed to be used by a single user, and is not designed to be used by multiple users.

Disclaimer

I made all efforts 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.

Table 1: Risk Classification

Likelihood	Level	High	Medium	Low
High		H	H/M	M
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

Scope

```
./src/
#-- PasswordStore.sol
```

Roles

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

Executive Summary

Issues found

Table 2: Issues Found

Severity	Number of issues found
High	2
Medium	0
Low	0
Info	1
Total	3

Findings

High

rivate **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 private variable and only accessable through `PasswordStore::getPassword` function , which is intended to be only called by the owner of the contract

We show one such method of reading data off chain below

Impact: Anyone can break the private password , severely breaking the function of the protocol

Proof of Concept: (Proof of Code) The below testcase shows how anyone can read the password directly from the blockchain.

Recommended Mitigation: The current logic of the protocol doesn't serve the intended functionality . So the logic needs to be rethought or worked upon . We can store the password off chain and then decrypt it ## LikeLihood & Impact : - Impact: High - Likelihood: High - Severity: High

High

- Worst offenders -> Least Bad

### [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 the owner to set a new password`

```
    //@audit any user can set the password
    //missing access control
    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 breaking the contract intended functionality.

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

Code

```
    function testAnyoneCanSetPassword(address randomAddress) public {
        vm.assume(randomAddress!=owner);
        vm.prank(randomAddress);
        string memory expectedPassword = "myNewPassword";
        store.setPassword(expectedPassword);
        vm.prank(owner);
        string memory actualPassword = store.getPassword();
        assertEq(actualPassword,expectedPassword);
    }
```

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

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

LikeLihood & Impact :

- Impact: High
- Likelihood: High
- Severity: High

Informational

[I-1] The `PasswordStore:getPassword` natspec indicates a parameter that doesn't exist causing the 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: Remove the natspec line.

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