Protocol Audit Report

Version 1.0

Cyfrin Updraft

June 6, 2025

Protocol Audit Report

Daniel Simanullang

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
  - Scope
  - Roles
- Executive Summary
  - Issues found
- Findings
- High
  - [H-1] the `s_password` variable is visible on-chain
  - [H-2] Non-owner can set the owner's password through `setPassword`
- Informational
  - [I-1] Incorrect natspec in `getPassword`

Protocol Summary

The `PasswordStore` protocol is a single-user protocol that enables the owner to store their password on-chain. The protocol does not support multiple users usage.

Disclaimer

The Daniel Simanullang 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
Likelihood	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

Executive Summary

Issues found

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

Findings

High

[H-1] the `s_password` variable is visible on-chain

Description Variables that are stored on-chain, despite them being `private`, is accessible to anyone that wish to view them. The intended way to view the `s_password` variable is trough the `getPassword` function. **Impact** Anyone can view the passwords stored on-chain. **Proof of Concepts**
1. Create a locally running chain

```
1     make anvil
```

2. Deploy the contract

```
1     make deploy
```

3. cast the storage tool

```
1     cast storage
```

4. cast the string parser to retrieve the password

```
1     cast parse-bytes32-string <hex>
```

Recommended mitigation Encrypt the password off-chain, and only store the encrypted password on-chain. Make sure to also not accidentally trigger a function that decrypts the user password

[H-2] Non-owner can set the owner's password through `setPassword`

Description The `setPassword` function lacks accessibility control, therefore making practically anyone to set everyone's password.

```
    //  @audit only the owner has to be able to set the password!
    function setPassword(string memory newPassword) external {
        s_password = newPassword;
        emit SetNewPassword();
    }
```

Impact Non-owners can set/change the password of the contract, severly breaking the contract intended functionality.

Proof of Concepts

```
    function test_non_owner_can_set_password(address randomAddress)
       public{
            vm.assume(owner!=randomAddress);
            vm.prank(randomAddress);
            string memory expectedPassword = "myPassword";
            passwordStore.setPassword(expectedPassword);

            vm.prank(owner);
            string memory actualPassword = passwordStore.getPassword();
            assertEq(actualPassword, expectedPassword);

        }
```

Recommended mitigation Add access control conditional to the `setPassword` function.

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

Informational

[I-1] Incorrect natspec in `getPassword`

Description The `function getPassword()` has no parameter, while the natspec indicates that it should be `function getPassword(string)`

Recommended mitigation Remove the following comment:

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