Protocol Audit Report

Version 1.0

41swara.com

January 15, 2025

PasswordStore Audit Report

41Swara

January 8, 2024

Prepared by: 41Swara

Lead Security Researcher: - Mrima Moses

Table Of Contents

Disclaimer

The 41Swara 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

The findings described in this document correspond the following commit hash:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

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

Protocol Summary

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

should be able to set and access this password.

Roles

- Owner: Is the only one who should be able to set and access the password.

For this contract, only the owner should be able to interact with the contract.

Executive Summary

Issues found

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

Findings

High

[H-1] Data On-Chain Is Not Private, Passwords Are Visible to Anyone

Description: Data stored on-chain is accessible and readable by anyone. Hence, you should not store private data that you do not intend to expose to the general public.

Impact: Anyone can see users' passwords, breaking the requirement that only users should be able to read their passwords.

Proof of Concept:

1. Run Anvil:

```
make anvil
```

2. Deploy the contract and copy the resulting contract address:

```
make deploy
```

3. Read the storage variable:

```
cast storage 0x5FbDB2315678afecb367f032d93F642f64180aa3 1

0x6d7950617373776f726400000000000000000000000000000000000000000014
```

4. Parse the hex data to a string:

```
cast parse-bytes32-string 0
   x6d7950617373776f726400000000000000000000000000000000000000000014


myPassword
```

Recommendations: Encrypt the password off-chain and store the encrypted password on-chain instead of storing the plaintext password.

[H-2] Missing Access Controls in `PasswordStore:setPassword`, Allowing Anyone to Set the Password

Description: The `PasswordStore:setPassword` function is set to `external`. According to the NatSpec, this function is supposed to be called by the owner only to set the password. However, there are no modifiers or checks to enforce this.

```
function setPassword(string memory newPassword) external {
  //@> No access controls
  s_password = newPassword;
  emit SetNetPassword();
}
```

Impact: Anyone can set or change the password, severely breaking the contract's intended functionality.

Proof of Concept:

Code

```
function testAnyoneCanSetPassword(address _randUser) public {
    string memory newPass = "newPassword";
    vm.prank(_randUser);
    passwordStore.setPassword(newPass);

    vm.prank(owner);
    string memory oldPass = passwordStore.getPassword();
    assertEq(newPass, oldPass);
}
```

Recommendations: Add a check to ensure only the owner can call this function.

Code

```
function setPassword(string memory newPassword) external {
+   if (msg.sender != owner) {
+       revert PasswordStore__NotOwner();
+   }
    s_password = newPassword;
    emit SetNetPassword();
}
```