Protocol Audit Report

Version 1.0

github.com/samigabor

April 26, 2024

Protocol Audit Report

Sami Gabor

April 26, 2024

Prepared by: Sami Gabor Lead Security Reasearcher: - Sami Gabor

Table of Contents

Protocol Summary

PasswordStore is a protocol for storage and retrieval of a user's password. The protocol is designed to be used by a single user. Only the owner should be able to set and access this password.

Disclaimer

I made all effort to find as many vulnerabilities in the code in the given time period, but hold 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

The findings in this report correspond to the following commit hash:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

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

Roles

- owner: the user who can set and read the password
- outsiders: no one else should be able to set or read the password

Executive Summary

Spent X hours using Y tools etc

Issues found

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

Findings

High

[H-1] Storing the password on-chain makes it visible to anyone and no longer private

Description: All data stored on-chin 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 the `PasswordStore::getPassword` function, which is intended to be only called by the owner of the contract.

Will show one such method of reading any data off chain below.

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

Proof of Concept: (proof of core) The below test case shows how you can read the private state variable from the blockchain: 1. run a local chain `anvil` 2. deploy the contract locally `make deploy` 3. read second storage slot `cast storage <contract-address> 1` 4. convert the output from hex to string `cast parse-bytes32-string <output>`

Recommended Mitigation:

The overall architecture of the protocol should be retaught.

One could encrypt the password off-chain, and the store the encrypted password on-chin.

This would require the user to remeber 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 sent a tx with the password that decrypts your password.

Or encrypt it with the user's public key and decrypt it with the user's private key.

[H-2] `PasswordStore::setPassword` is missing the access control, meaning non-owner can change the password

Description: `PasswordStore::setPassword` external function doesn't restrict access to the owner of the contract. This means that anyone can change the password, which is a critical vulnerability.

```
    function setPassword(string memory newPassword) external {
@>      // @audit missing access control
        s_password = newPassword;
        emit SetNetPassword();
    }
```

Impact: Anyone can change the password, severely breaking the intended functionality of the contract.

Proof of Concept: (proof of core)

Code

```
    function test_anyone_can_change_password(address randomAddress) public {
        vm.assume(randomAddress != owner);

        string memory expectedPassword = "myChangedPassword";
        vm.prank(randomAddress);
        passwordStore.setPassword(expectedPassword);

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

Recommended Mitigation: Add access control to `PasswordStore::setPassword`

```
    if(msg.sender != owner) revert PasswordStore__NotOwner();
```

Informational

[I-1] Incorrect natspec for `PasswordStore::getPassword`

Description: `PasswordStore::getPassword` natspec declares a `newPassword` which doesn't exist.

Impact: Incorrect natspec

Recommended Mitigation: Remove the `newPassword` param from the natspec

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