Protocol Audit Report

Version 1.0

equious.eth

March 1, 2024

Protocol Audit Report

Cyfrin-Malik

March 1, 2024

Prepared by: Cyfrin Lead Security Researcher:

- Malik

Table of Contents

- Informational
  - [I-1] Documentation mentions a parameter when there is none present, meaning the natspec is incorrect
  - Likelihood & Impact:

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.

Disclaimer

The Shortcut-23 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

Commit hash

```
- Commit Hash: 2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

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

Roles

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

Executive Summary

We spent 72 hours on learning and writing this file however the audit was done under 2 hours of work. Used VSCode, Foundry and solidity code metrics.

Issues found

Severity	Issues Found
High	2
Medium	0
Low	0
Info	1
—	—
Total	3

Findings

High

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

Description: All data stored on chain is visible to anyone, the `PasswordStor::s_password` is intended to be private and meant to be only accessible by the `PasswordStore::getPassword()` function, accessible only to the owner.

One such method of reading private set variables is shown below.

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

Proof of Concept: (Proof of code) The below test showcases how anyone can read a private variable directly off the blockchain.

1. deploy a testnet: `make anvil`
2. deploy the contract: `make deploy`
3. read the variables: `cast storage <contract address from terminal of the deployed contract>`
4. retrieve the storage variable in bytes and then convert it to a string: `cast parse-bytes32 -string <variable in bytes of relevant variable from previous output>`

result string is the variable that was supposed to be private; `myPassword`

Recommended Mitigation: //edit the text below in the morning the whole protocol to store it this way does not work as wanted. password encrypted off chain should be stored and then deployed in the contract and the view function would also not be suitable at all.

Likelihood & Impact:

Severe disruption of functionality or availability: Yes

- Impact: HIGH
- Likelihood: HIGH
- Severity: HIGH

[H-2] `PasswordStore::setPassword(...)` has no limit to who can run it, meaning a non-owner can change it

Description: The comments suggest that only the owner could access the function and the function is set as external. * `@notice This function allows only the owner to set a new password`. However there is no limitation in the function set such as an `onlyOwner` modifier or a revert if nonOwner or any if statement that checks first if the caller of the function is the owner. I.e. access control is missing.

```
function setPassword(string memory newPassword) external {
        s_password = newPassword;
        emit SetNetPassword();
    }
```

Impact: Anyone can set or change the password of the contract, severly breaking the intended functionality of the contract.

Proof of Concept: A test has been made to proof that anyone can set the password. Add the following to the `PasswordStore.t.sol` (and as best practice this filename should be `PasswordStoreTest.t.sol`)

Code

```
function test_non_owner_can_set_password(address randomAddress) public
    {
        vm.assume(randomAddress != owner); //exclude the owner from the
             random generated addressess
        vm.prank(randomAddress); // act as the randomAddress
        string memory newPassword = "proof anyone can set a password";
            //set a new password
        passwordStore.setPassword(newPassword); //run the function with
             the newPassword as parameter
        //some random Address set the password

        vm.prank(owner); // be the owner in the next line to retrieve
            the password
        string memory actualPassword = passwordStore.getPassword();
        assertEq(newPassword, actualPassword); //check that the set
            password by the randomAddress is the one that the owner
            retrieves
    }
```

Recommended Mitigation: Add a restriction/access control functionality to the function `PasswordStore::setFunction` such as:

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

Since this would be the second time it is set, perhaps this line could be in a modifier and added to the two functions which is tidier.

Informational

[I-1] Documentation mentions a parameter when there is none present, meaning the natspec is incorrect

Description:

```

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

    //  @audit there is no newPassword parameter

    function getPassword() external view returns (string memory) {
        ...
    }
```

The `PasswordStore::getPassword()` function signature is `getPassword`, natspec mentions `getPassword(newPassword)`.

Impact: The natspec is incorrect

Recommended Mitigation: Remove the incorrect natspec line.

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

Likelihood & Impact:

- Impact: NONE
- Likelihood: NONE
- Severity: Information/Gas/Non-crits