Protocol Audit Report

Version 1.0

Cyfrin.io

April 22, 2024

PassWordStore Report

QV

April 22, 2023

Prepared by: QV Lead Auditors: - xxxxxxx

Table of Contents

Protocol Summary

PasswordStore is a smart contract applicatoin for storing a password. Users should be able to store a password and then retrieve it later. Others should not be able to access the password.

Disclaimer

The YOUR_NAME_HERE 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:

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

Scope

In Scope:

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

- Solc Version: 0.8.18
- Chain(s) to deploy contract to: Ethereum

Roles

Executive Summary

Issues found

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

Findings

High

[H-1] State variable is stored on-chain - the password is not private

Description: All on-chain data can be read by anyone, the `PasswordStore::s_password` variable is intended to be private to the owner and can only accessed through the `PasswordStore::getPassword` function, which is only called by the contract's owner.

We show on such method of reading any data off chain below. **Impact:** Anyone can read the private password, severly breaking the functionality fo the protocol

Proof of Concept:

See the test case below

1. Create a locally running chain

```
make anvil
```

2. Deploy the contract to the chain

```
make deploye
```

3. Run the storage tool

Access the 1 slot on the storage that associate with `s_password` state variable

```
cast parse-bytes32-string <CONTRACT_ADDRESS> 1 --rpc-url http://
    127.0.0.1:8545
```

You will get the result looks like this: `0x6d7950617373776f726400000000000000000000000000000000000000000014`

Then you can parse that hex to string value and get the out put `myPassword`

```
cast parse-bytes32-string 0
    x6d7950617373776f726400000000000000000000000000000000000000000014
myPassword
```

Recommended Mitigation: Review the system design.

[H-2] Access control - anyone can set the password

Description: `PasswordStore:: setPassword` is set as an `external` function whic leads to access control issue.However, according the the documentation `This fuction allows only the ownver to set a new password`

```
    function getPassword() external view returns (string memory) {
        if (msg.sender != s_owner) {
            revert PasswordStore__NotOwner();
        }
        return s_password;
    }
```

Impact: Anyone can set/change the password of the contract which break the intended function of the contract.

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

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

Recommended Mitigation: Add access control condition to the `setPassword` function

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

Informational

Description: The `PasswordStore::getPassword` function does not require a parameter, contrary to what the natspec documentation states. It incorrectly indicates that the function should accept a parameter `newPassword`.

Impact: The natspec documentation is misleading and may cause confusion for developers implementing or interacting with the contract.

Proof of Concept: The discrepancy between the natspec documentation and the actual function signature can be observed directly in the source code.

Recommended Mitigation: Remove the incorrect natspec line specifying `@param newPassword` from the documentation to align it with the actual function signature.

“‘diff - @param newPassword The new password to set.