Protocol Audit Report

Version 1.0

Cyfrin.io

February 8, 2024

Passwordstore Audit Report

tobez

9 feb, 2024

Prepared by: Tobez Lead Auditors:

- alone

Table of Contents

Protocol Summary

PasswordStore is protocol dedicated to storing password and retrieving it. 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 tobez's 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: 2e8f81e263b3a9d18fab4fb5c46805ffc10a9990

Scope

- In Scope:

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

Roles

Outsider: Should not have any permissions Owner: Should be able to create and get password stored by them alone

Executive Summary

Issues found

security	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 chain are visible to anyone, and can be read from the blockchain. The `PasswordStore::s_password` variable is intended to be private and can be accessed only by the owner via the `PasswordStore::getPassword` function which should be called only by the owner.

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

Proof of Concept: The below test case shows how anyone can access the password from the blockchain

1. create a local blockchain

```
make anvil
```

2. Deploy your contract

```
make deploy
```

3. Use the following command to access the storage

```
cast storage <CONTRACT_ADDRESS> 1 --rpc-url http://127.0.0.1:8545
```

it will return a bytes string 0x6d7950617373776f726400000000000000000000000000000000000000000014

4. convert from bytes to string

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

you get the following string myPassword

Recommended Mitigation: The architecture of the smart contract needs to be redo, storing passwords via an off-chain platform and then storing it encrypt on-chain where the owner have to remember off-chain password to get the main password is a better and more secure solution. Also removing the view function from the get password

[H-2] `PasswordStore::setPassword` has no access control; non-owner can change the password

Description: The `PasswordStore::setPassword` method is set to external function, however,the natspec of the function and the purpose of the contract s to allow `Only owner f a password to accept it; no everyone`

```
function setPassword(string memory newPassword) external { //public
        //@audit it is missing access control
        s_password = newPassword;// anyone can call this function not
            only owner, create a modifier or use if statement
        emit SetNetPassword();
    }
```

Impact: Anyone can call this function and change the password, which makes the system impossible to use

Proof of Concept: Add a new test function that pull random address to set password

code

```
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 an access control with a simple IF CONDITION with good error

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

Informational

[I-1] The `PasswordSore::getPassword()` method natspec suggests that there should be a parameter but there is none

Description: The `PasswordSore::getPassword()` method signature is `getPassword()` but natspec said it should be `getPassword(string)`

Impact: The natspec is incorrect

Recommended Mitigation: Remove the incorrect natspec

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