Protocol Audit Report

Version 1.0

Rutvik

December 16, 2023

Protocol Audit Report

Rutvik Gujarati

December 16, 2023

Prepared by: [Rutvik Gujarati] Lead Auditors: - xxxxxxx

Table of Contents

- Informational
  - [I-1] The `passwordStore:: getPassword()` indicates that doesn't exists, it causing the netspec to be incorrect
  - Impact and Likelihood:

Protocol Summary

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

I 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 me 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

Finding described in this document correspond with commit hash:

- Commit Hash: 2e8f81e263b3a9d18fab4fb5c46805ffc10a9990 ## Scope

--> PasswordStore.sol

Roles

Owner: who can set the password and read the password. Outsiders: no one else can be able to set and read the password.

Executive Summary

–> Learnign Phase of Auditing with Cyfrin Audit Course(Patric Collains)

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 ON-CHAIN USING PRIVATE KEYWORD IS VISIBLE TO EVERYONE, IT IS NO LONGER PRIVATE FOR ONLY OWNER.

Description: Data that stored on on-chain it is publically acceseble no metter that has visibility provided by contract. It only worked for contract not for humans readability.

The `PasswordStore::s_password` vaiable is intented to a private variable and only access through `PasswordStore::getpassword()` function, which should be only call by only owner.

Impact: Anyonne can read the Private password, severly breaking the fucntionality of the protocol.

Proof of Concept: (proof of code)

1. create a local running chain `make anvil`
2. deploy contract locally `make deploy`
3. run the storge tool

use 1 because `s_password` is in second storage slot

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

Get output like this: `0x436f6e74726163742d506173730000000000000000000000000000000000001a`

now you can parse to string using:

`cast parse-bytes32-string 0x436f6e74726163742d506173730000000000000000000000000000000000001a`

output: `Contract-pass` which is the user password.

Recommended Mitigation: One could encrypt the password off-chain and then store encrypted password on-cahin. It could require user to remember another password off-chain to decrypt the password. Remove view fucntion for user to accedentilly send a transaction with the password that decrypt your password.

Impact and Likelihood:

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

[H-2] `passwordStore:: Setpassword()` has no access , anyone can change the password.

Description: This `passwordStore:: Setpassword()` function is set to be an `external` function however, This function has no allowance of `ownership`.

```
function setPassword(string memory newPassword) external {
@>    //@Audit - There are no access controll checking
        s_password = newPassword;
        emit SetNetPassword();
    }
```

Impact: Anyone can set/change the conrtact password. that may break the contract functionality.

Proof of Concept: add below test code into the `passwordStore.t.sol`

Code

```
    function anyone_can_set_password(address randomAddr)public  {
         vm.assume(randomAddr != owner);
         vm.prank(randomAddr);
         string memory ExpectPass = "MyPassword";
         passwordStore.setPassword(ExpectPass);

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

Recommended Mitigation: Add an additional access control in `PasswordStore:: setPassword()`

```
if(msg.sender!= owner){
    rever Error;
}
```

Impact and Likelihood:

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

Informational

[I-1] The `passwordStore:: getPassword()` indicates that doesn't exists, it causing the netspec to be incorrect

Description:

```
//  @param newPassword The new password to set.
    function getPassword() external view returns (string memory) {
```

`The `passwordStore:: getPassword()` function signature is getpassword() which netspec say it should be `getPassword(string)`

Impact: The netspec is incorrect.

Recommended Mitigation: Remove the incorrect netspec line

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

Impact and Likelihood:

- Impact: NONE
- Likelihood: HIGH
- Severity: Informational/Gas/Non-Crits