Protocol Audit Report

Version 1.0

Cyfrin.io

December 6, 2023

Protocol Audit Report

radin100

December 6, 2023

Prepared by: radin100 Lead Security Researcher- Table of Contents - Table of Contents - Protocol Summary - Disclaimer - Risk Classification - Audit Details - Scope - Roles - Executive Summary - Issues found - Findings - High - [H-1] TITLE Storing the password on-chain makes in visible to anyone, and no longer private - Likelihood & Impact - [H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner can change the password - Likelihood & Impact - Informational - [I-1] `PasswordStore::getPassword` has missing a parameter - Likelihood & Impact

Table of Contents

        * [H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner can change the password
    - Likelihood & Impact
- Informational
    - [I-1] `PasswordStore::getPassword` has missing a parameter
    - Likelihood & Impact

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

The Radin Radev 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 him 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

Scope

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

Roles

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

Executive Summary

This protocol took 1 hour to 1 auditor to be completely audited.

Issues found

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

Findings

High

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

Description: All data stored on-chain 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::s_password` function, which is intended to be only called by the owner of the contract

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

Proof of Concept: (Proof of Concept)

1. Create locally running chain

```
make anvil
```

2. Deploy the contract

```
make deploy
```

3. Run the storage tool

Get the password ref by using cast storage and get the element number 1 from it.

```
cast storage 0x5FbDB2315678afecb367f032d93F642f64180aa3 1 --rpc-url
    http://127.0.0.1:8545
```

Then parse the returned bytes32 value to string by using cast parse-bytes32-string

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

Recommended Mitigation: You should see another way of getting `PasswordStore::s_password` more secure.

Likelihood & Impact

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

[H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner can change the password

Description: The `PasswordStore::setPassword` function is set to be an `external` function, that will not consider the funcionality `This function allows only the owner to set a new password`

```
function setPassword(string memory newPassword) external {
        //@audit there are no access cotrol
        s_password = newPassword;
        emit SetNetPassword();
    }
```

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

Proof of Concept: Add the following to the `PasswordStore.t.sol` 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 an access conditional to the `PasswordStore::setPassword` function.

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

Likelihood & Impact

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

Informational

[I-1] `PasswordStore::getPassword` has missing a parameter

Description: `PasswordStore::getPassword` does not accept any string type which is required functionality form the documentation.

Impact: As calling `PasswordStore::getPassword` the `PasswordStore::s_password` won't be changed.

Recommended Mitigation:

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

Likelihood & Impact

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

Information: This function is missing some functionality.