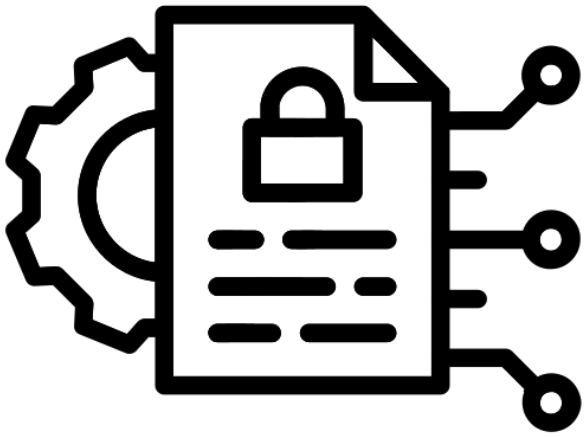

Protocol Audit Report

Version 1.0

Cyfrin.io

November 15, 2024

Protocol Audit Report

Eduardo Melo

Oct 25, 2023

Prepared by: [Eduardo] Lead Auditors: Security Researcher - Eduardo

Table of Contents

- Informational
  - [I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn´t exist, causing the natspec to be incorrect.
  - Likelihood & Impact:
- Gas

Protocol Summary

Disclaimer

The Eduardo´s Team 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
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

Commit Hash:

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

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

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-chain makes it visable to anyone, and no longer private

Description Variables stored in storage on-chain are visable to anyone, no matter the solidity visibility keyword meaning the password is not actually a private password

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

Proof of Concept

1. Start a local node

```
make anvil
```

2. Deploy

This will default to your local node. You need to have it running in another terminal in order for it to deploy.

```
make deploy
```

3. Run the storage tool

we use 1 because that´s the storage slot of `s_password` in the contract.

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

You´ll get an output that looks like this:

`0x6d7950617373776f726400000000000000000000000000000000000000000014`

You can then parse that hex to a string with.

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

Recommended Mitigation Due to this, the overall archtecture of the contract should be rethought. This would require the user to remember another password off-chain , and then store the encrypted password on-chain. This would require the user to remember another password off-chain to decrypt the password. However, you´d also likely want to remove the view function as you wouldn´twant to user to accidentally send a transaction with the password that decrypts your password. ### Likelihood & Impact: - Impact: HIGH - Likelihood: HIGH - Severity: HIGH

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

Description: The `PasswordStore::setPassword` function is set to be an `external` function, however, the natspec of the function and overall purpose of the smart contract is that `This function allows only the owner to set a new password.`

```
function setPassword(string memory newPassword) external {
@>      // @audit - There are no access controls
        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` test file.

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 conditional to the `setPassword` function.

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

Likelihood & Impact:

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

Medium

Low

Informational

[I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn´t exist, causing the natspec to be incorrect.

Description

```
/*
     * @notice This allows only the owner to retrieve the password.
     * @param newPassword The new password to set.
     */
    function getPassword() external view returns (string memory) {
```

The `PasswordStore::getPassword` function signature is `getPassword()` which the natspec say it should be `getPassword(string)`

Impact The natspec is incorrect.

Recommended Mitigation Remove the incorrect natspec line.

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

Likelihood & Impact:

Gas