Protocol Audit Report

Version 1.0

Emillystev

October 9, 2024

Protocol Audit Report

Emillystev

October 9, 2024

Prepared by: [Emillystev] Lead security researcher: - xxxxxxx

Table of Contents

Protocol Summary

PasswordStore is a protocol dedicated to storage and retrieval of a user's password. The protocol is designed to be used by a single user, and is not designed to be used by multiple users. Only the owner should be able to sett and access this password.

Disclaimer

Emillystev 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

*** The findings described in this document respond the following commit hash: ***

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

Scope

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

Roles

owner: the user who can set the password read the password outsiders: no one else should be able to set or read the password

Executive Summary

we spend X hours with Z auditors using Y tools

Issues found

Findings

High

[H-1]storing the password on-chain makes it visible to anyone and no longer private

*** Description:** All data stored on-chain is visible to anyone and can read directly from the blockchain. the `PasswordStore::s_password` variable is intended to be a private variable and only accessed through the `PasswordStore::getPassword` function, which is intended to be only callable by the owner of the contract

we show one such method of reading any data off chain below

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

Proof of Concept: (Proof of code) the below test case shows how anyone can read the password directly from the blockchain

Recommended Mitigation: due to this, overall architecture of the contract should be rethought. one could encrypt the 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't want the user to accidentally send a transaction with the password that decrypts your password.

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

[H-2] `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 {
    >@    //  - there are no access control
        s_password = newPassword;
        emit SetNetPassword();
    }
```

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

Proof of Concept: (Proof of code) 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 conditionalto the ‘setPassword’ function

Code

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

Medium

Low

Informational

[I-1] The PasswordStore::getPassword natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect

*** Description: ***

Code

```
1
2       /*
3        * @notice This allows only the owner to retrieve the password.
4    @>  * @param newPassword The new password to set.
5        */
6       function getPassword() external view returns (string memory) {
```

The natspec for the function PasswordStore::getPassword indicates it should have a parameter with the signature getPassword(string). However, the actual function signature is getPassword().

Impact: The natspec is incorrect.

Recommended Mitigation: Remove the incorrect natspec line.

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