PasswordStore Protocol Audit Report

Version 1.0

Aditya Pratap Singh

December 24, 2024

PasswordStore Protocol Audit Report

Aditya Pratap Singh

Dec 24, 2024

Prepared by: Aditya Pratap Singh Lead Auditors: - Aditya Pratap Singh

Table of Contents

Protocol Summary

A smart contract application 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 Aditya Pratap Singh 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

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

- Solc Version: 0.8.18
- Chain(s) to deploy contract to: Ethereum

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

Spent 1 hour reviewing the codebase and found 1 High and 1 informational bug.

Issues found

Severity	Number of Issues Found
High	2
Medium	0
Low	0
Info	1
Gas	0
————	——————————
Total	3

Findings

High

[H-1] Storing password on-chain makes it visible, and so no longer private

Description All the data stored on-chain is public and can be read directly from the blockchain. The `PasswordStore::s_password` is intented to be private and only accessed with the function `PasswordStore::get_password` function, intented to be called by the owner of the contract.

Impact Anyone can read the password.

Proof of Concept (Proof of code)

The below testcase shows how anyone can read the password on-chain

```
make anvil
```

```
make deploy
```

Run the storage tool

```
cast storage <Address> 1 -- --rpc-url <RPC_URL>
```

The we get the password by parsing the hex value

```
cast parse-bytes32-string <Hex_Value>
```

we get the password

Recommended Mitigation Due to this, the 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 accidently send a transaction send a transaction with the password that decrypts your password.

[H-1] `PasswordStore::set_password` function has non-owner access control so anyone can change the password.

Description The overall purpose of the `PasswordStore::set_password` is that it should allows only owner of the contract can change the password. However it is set as an `external` function.

```
function setPassword(string memory newPassword) external {
    // @audit -> no access controls
    s_password = newPassword;
    emit SetNetPassword();
}
```

Impact Any non-owner can change the password. severaly breaking the functionality of the contract.

Proof of Concept (Proof of code)

The below testcase shows how anyone can set the password.

Code

```
function test_non_owner_can_change_password() public {
    vm.startPrank(address(1));
    string memory expectedPassword = "newPassword";
    passwordStore.setPassword(expectedPassword);

    vm.startPrank(owner);
    string memory ownerPassword = passwordStore.getPassword();

    assertEq(expectedPassword, ownerPassword);
}
```

Recommended Mitigation Add an access control conditional to the `setPassword` function.

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

Informational

[I-1] The `PasswordStore::getPassword` function signature is `getPassword()` while the natspec says it should be `getPassword(string)`

Impact The natspec is incorrect

Recommended Mitigation Remove the natspec line.

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