Protocol Audit Report

Version 1.0

Operation-C

September 25, 2024

PasswordStore Audit Report

Operation-C

Sept 25, 2024

Prepared by: Operation-C

Lead Security Reacher: Operation-C

Table of Contents

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 Operation-C 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

Scope

```
./src/
#-- Passwordstore.sol
```

Roles

- Owner: The user who can set the passwrod 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
Gas	0

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 public and visible to anyone. The `PasswordStore::s_password` variable is intended to be hidden and only accessible by the owner through the `PasswordStore::getPassword` function.

I show one such method of reading any data off chain below.

Impact: Anyone is able to read the private password, severly breaking the functionality of the protocol.

Proof of Concept:

1. Create a locally running chain

```
make anvil
```

2. Deploy the contract to the chain

```
make deploy
```

3. Run the storage tool

```
cast storage <CONTRACT ADDRESS> 1 --rpc-url http://localhost: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
```

And then get an output of:

```
myPassword
```

[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 purpose of the smart contract and function's natspec indicate 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 SetNewPassword();
}
```

Impact: Anyone can set/change the stored password, severly breaking the contract's intended functionality

Proof of Concept: Add the following to the PasswordStore.t.sol test file:

```
    function test_access_control_of_set_password(address randomAddress)
        public {

        vm.prank(randomAddress);
        string memory newPassword = "newPassword";
        passwordStore.setPassword(newPassword);

        vm.prank(owner);
```

```
8           string memory actualPassword = passwordStore.getPassword();
9
10          assertEq(actualPassword, newPassword);
11      }
```

Recommended Mitigation: Add an access control conditional to `PasswordStore::setPassword`.

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

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()` while the natspec says it should be `getPassword(string)`.

Impact: The natspec is incorrect

Recommended Mitigation: Remove the incorrect natspec line.

```
/*
* @notice This allows only the owner to retrieve the password.
-> * @param newPassword The new password to set.
*/
```