First Basic Audit Report

Version 1.0

Jarrod Pyne

April 26, 2024

First Basic Audit Report

Jarrod Pyne

26 April, 2024

Prepared by: Jarrod Pyne

Table of Contents

Protocol Summary

PasswordStore is a protocol that stores and retrieves a user’s passwords. It is designed for a single user and as such, is only able to be set and accessed by the specified owner.

Disclaimer

Jarrod Pyne 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

The findings described in this document correspond the following commitment 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

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

Issues found

Findings

High

[H-1] Variables stored in storage on-chain are visible to anyone

Description

All data stored on-chain is visible to anyone. As a result, it can be read directly from the blockchain. In this code, the `PasswordStore::s_password`, despite being marked as `private` is still capable of being viewed. It is my understanding this variable is only to be viewable by the owner.

Impact

Anyone can read the password, severely breaking the protocol.

Proof of Concepts

Below test shows how anyone can read from the blockchain:

1. Create a locally running chain

```
make anvil
```

2. Deploy the contract to the chain

```
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 0x6d7950617373776f726400000000000000000000000000000000000000000014
```

And get an output of:

```
myPassword
```

Recommended mitigation

The architecture of the protocol may need to be re-thought. Provided a password is unable to be stored on-chain, please consider either: 1. storage of the password off-chain (i.e. hard copy of the written password); or 2. encrypting the password, off-chain and including the password in the protocol.

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

Description The `PasswordStore::setPassword` function is set to an `external` function that is capable of being called by anyone. The documentation outlines that the purpose of this function is that only the owner is capable of changing this password.

```
    // @audit there needs to be an onlyOwner modifier
    function setPassword(string memory newPassword) external {
        s_password = newPassword;
        emit SetNetPassword();
    }
```

Impact Anyone can set the password, severely breaking the protocol.

Proof of Concepts

```
    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 mitigation to the `setPassword` function.

```

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

Informational

[I-1] `PasswordStore::getPassword` function does not have any specified parameters

Description

The `PasswordStore::getPassword` function does not specify any parameters.

See below:

```
  // @audit this doesn't have a parameter

   --> function getPassword() external view returns (string memory) {
        if (msg.sender != s_owner) {
            revert PasswordStore__NotOwner();
        }
        return s_password;
    }
```

Impact The effect of this means that the nat spec in the documentation is incorrect.

Recommended mitigation

Please consider removing this:

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