Protocol Audit Report

Version 1.0

Cyfrin.io

October 22, 2024

3-Password-Store Protocol Audit Report

Brandon Norman

October 22, 2023

Prepared by: [Tiger Audits] Lead Auditors: - Brandon Norman

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
  - Scope
  - Roles
- Executive Summary
  - Issues found
- Findings
- High
  - [H-1] Storing the Password On-Chain Makes It Visible to Anyone, and No Longer Private
  - [H-2] `PasswordStore::setPassword` Has No Access Controls, Allowing Non-Owners to Change the Password
- Informational
  - [I-1] The `PasswordStore::getPassword` Natspec Indicates a Parameter That Doesn't Exist, Causing Incorrect Documentation

Protocol Summary

3-Password-Store protocol stores 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 Tiger Audits 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 ## 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 two hours with one auditor using 3 tools. ## Issues found | severity | Number of issues found | | ————- | ———————- | | Highs | 2 | | Medium | 0 | | Low | 0 | | Informational | 1 | | Total | 3 |

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 be read directly from the blockchain. The `PasswordStore::s_password` variable is intended to be private and should only be accessed through the `PasswordStore::getPassword` function, which is meant to be called only by the owner of the contract.

We demonstrate one method of reading any data off-chain below.

Impact: Anyone can read the private password, severely compromising the functionality of the protocol.

Proof of Concept: The following test case shows how anyone can read the password directly from the blockchain:

1. Create a locally running chain: `bash make anvil`
2. Deploy the contract to the chain: `bash make deploy`
3. Run the storage tool: We use 1 because that's the storage slot of `s_password` in the contract. `bash cast storage <ADDRESS_HERE> --rpc-url http://127.0.0.1:8545`

   You will receive output that looks like this:

   ```
   0x6d7950617373776f726400000000000000000000000000000000000000000014
   ```

   You can then parse that hex into a string with:

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

   The output will be:

   ```
   myPassword
   ```

Recommended Mitigation:
The overall architecture of the contract should be reconsidered. One approach is to encrypt the password off-chain and store the encrypted password on-chain. This would require users to remember another password off-chain to decrypt the password. Additionally, you may want to remove the view function to prevent users from accidentally sending a transaction that reveals the password used for decryption.

[H-2] `PasswordStore::setPassword` Has No Access Controls, Allowing Non-Owners to Change the Password

Description: The `PasswordStore::setPassword` function is marked as `external`. However, the natspec of the function states that it is intended to allow 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 or change the password of the contract, severely breaking 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 condition to the `setPassword` function:

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

Informational

[I-1] The `PasswordStore::getPassword` Natspec Indicates a Parameter That Doesn't Exist, Causing Incorrect Documentation

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) {
    // function implementation
}
```

The `PasswordStore::getPassword` function signature is `getPassword()`, but the natspec incorrectly states that it should accept a `string` parameter.

Impact: The natspec documentation is incorrect.

Recommended Mitigation:
Update the natspec to remove the erroneous parameter:

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