E.A RESEARCH
SECURING SMART
CONTRACTS

PasswordStore Protocol Audit Report

Version 1.0

E.A Research

July 11, 2024

PasswordStore Protocol Audit Report

Emmanuel Acho, Phd (Thanks to Patrick Collins - Cyfrin Updraft)

July 10, 2024

Prepared by: Emmanuel Acho, PhD

Table of Contents

Protocol Summary

This protocol enables the owner of the contract to store and retrieve their password. It is designed to be used by a single as opposed to multiple users.

Disclaimer

E.A Research 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 are based on the following commit hash:

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

Scope

The scope of the audit is restricted to the following file in the `src` folder:

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

Roles

- Owner: The user who can set and read the password
- Outsiders: No one else should be able to set or read the password

Executive Summary

This audit was done as part of the Cyfrin Updraft Security Research Curriculum

Issues found

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

Findings

High

[H-1] Storing the password on-chain (Problem) makes it visible to anyone (Impact)

Likelihood & Impact

- Impact: High: There can be a severe disruption to protocol functionality
- Likelihood: High: No conditions have to be met for the vulnerability to be exploited
- Severity: High: There is a direct impact on the functionality of the protocol.

Description: All data stored on-chain is visible to anyone and can be directly read from the blockchain. The `PasswordStore::s_password` variable is intended to be a private variable only accessible through the `PasswordStore::getPassword` function which can only be called by the owner of the contract. An example (a method) of reading any data off chain is shown below.

Impact: Anyone can read the password which severely breaks the functionality of the protocol.

Proof of Concept The test case below shows how anyone can read the password directly 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 `1` is used because it is the storage slot of the `PasswordStore::s_password` variable in the contract.

```
cast storage <CONTRACT_ADDRESS> 1 --rpc-url http://127.0.0.1:8545
```

The output will look like: `0x6d7950617373776f7264000000000000000000000000000000000000000000`

It can then be parsed to a string with:

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

The output: `myPassword`

Recommended Mitigation You need to rethink the entire architecture of the contract with the following considerations: 1. Encrypyt the password offchain and store the encrypted password onchain. 1.1. This requires of the user to remember another password ofchain in order to decrypt the password. 2. You may also have to **remove the view function**. 2.1. This will prevent the user from accidentally sending a transaction with the password that decrypts their password.

Finding 2

[H-2] `PasswordStore::setPassword` has no access controls (Problem). A non-owner can change the password (impact)

Informational

[I-1] `PasswordStore::getPassword` natspec indicates a parameter that does not exist. This is incorrect documentation information.

Impact & Likelihood

- Impact: None - This is wrong documentation issue, there is no disruption to the protocols functionality
- Likelihood: N/A - There is no risk associated with this vulnerability
- Severity: Informational - Documentation should be fixed

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()` whereas the natspec suggets 'getPassword(string).

Impact: The natspec is incorrect

Recommended Mitigation:

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