Protocol Audit Report

Version 1.0

Cyfrin.io

July 3, 2024

Protocol Audit Report

Mohammad Arif Fikree

July 3, 2024

Prepared by: Mohammad Arif Fikree Lead Auditors: - Mohamamd arif Fikree

Table of Contents

Protocol Summary

PasswordStore is a protocol dedicated to storage and retrieval of a user's passwords. The protocol is designed to be used by a single user, and is not designed to be used by multiple users. Only the owner should be able to set and access this password.

Disclaimer

The Mohammad Arif Fikree 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

The findings described in this document correspond the following commit hash:

```
ff694529248699c59d991022108530247a92d56d
```

Scope

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

Roles

- Owner: The user who can set the password and read the password.
- Outsides: No one else should be able to set or read the password. # Executive Summary *Add some notes about how the audit went, types of things you found, ethc. We spent X hours with Z auditors using Y tools. etc* ## 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 makes it visible to anyone, and no longer private

Description All data on-chainis visible t anyone, and can be read dircetly from the blockchain. The `PasswordStore:: s_password` variable is intended to be a private variable and only accessed through the `PasswordStore:: getPassword` function, which is intended to be only called by the owner of the contract.

we show one such method of reading any data off chain below. **Impact**Anyone can read the private password, severly breaking the functionality of the protocal

Proof of Concepts(Proof of code) The below test case 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 We use 1 because that is the storage slot of `s_password` in the contract

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

You will get the output looks like this:

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

You can then parse that hex to a string with:

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

And get the output of:

```
myPassword
```

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 accidentally send a transaction with the password that decrypts your password.

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

Informational

[I-1] The `PasswordStore::getPassword` NatSpec Indicates a Parameter That Doesn't Exist

Description

The `PasswordStore::getPassword` function signature is as follows:

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

Impact The NatSpec documentation is incorrect, which can mislead developers and users of the contract.

Recommended mitigation Remove the incorrect natspec line.

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

Notes: The use of `NatSpec` (Ethereum Natural Specification Format) helps in generating accurate and helpful documentation. Ensuring that these comments match the actual function signatures and behaviors is crucial for maintaining clarity and correctness in the contract documentation.