Protocol Audit Report

Version 1.0

SADFrancis

September 28, 2024

PasswordStore Audit Report

Sean Francis

September 28th, 2024

Prepared by: Team SADFrancis Lead Auditors: - Sean Francis

Table of Contents

Protocol Summary

Protocol stores a password set by a the contract owner and users can call a function to retrieve the password. There is only one password stored per contract.

Disclaimer

The SADFrancis 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 with the following Commit 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

- I followed the Smart Contract Audit Youtube course, section 3, by Cyfrin Audits and Team Red Guild.

`https://www.youtube.com/watch?v=pUWmJ86X_do`

- This was an introduction to auditing; the basic work flow, and tools to write up the report. The tools used to find the example bugs were all built in foundry.

Issues found

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

Findings

High

[H-1] Private Variables are Still Publicly Available To Read Onchain: Passwords are not 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 a private variable and only accessed through the `PasswordStore::getPassword` function that intended be used by only the contract owner.

We show one such method fo reading any data off chain below.

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

Proof of Concept: (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 to obtain the contract address

```
make deploy
```

3. Run the storage tool, the 1 parameter is the second storage slot of the contract, `s_password`

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

You'll receive the output:

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

4. Parse the bytes data above into a string with the command:

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

The output will be:

```
myPassword
```

The same string found as the parameter to `PasswordStore::setPassword` function called in the `script/DeployPasswordStore.s.sol` script.

Recommended Mitigation: Due to the this, the overall architecture of the contract must be refactored. One could encrypt the password off-chain and the contract can be used to store the encrypted version on-chain. This would require the user to remember another password off-chain to decrypt the password.

- don't understand this line: 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 thar decrypts your password.

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

Informational

[I-1] `PasswordStore::getPassword` natspec indicates a paramter that does not exist causing the natspec to be incorrect.