Protocol Audit Report

Prepared by: Fish Scale

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
  - Scope
  - Roles
- Executive Summary
  - Issues found
  - | Info | 1 |
- Findings
  - High
    - [H-1] The password is stored on-chain, therefore its readable
    - [H-2] Missing access control, else anyone can set the password
  - Informational
    - [I-1] Incorrect use of Natspec as there is no use of param as intended in the Natspec

Protocol Summary

A smart contract application for storing 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 Fish Scale 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

The findings in this document correspond the following commit hash

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

Scope

```
./src/
└── PasswordStore.sol
```

Roles

- Owner: The user who can read and write the password
- Outsiders: No one is able to read or write the passwords

Executive Summary

add some notes about how the audit went, things you found, hours you spent and tools you used

Issues found

Severity	Number of Issues Found
High	2
Medium	0
Low	0
Info	1

Total 3

Findings

High

[H-1] The password is stored on-chain, therefore its readable

Description: All on-chain data is visible to anyone as it can be read via storage. As `PasswordStore::s_password` is intended to be private, yet it can be read as it's on-chain.

Impact: Anyone can read the private password.

Proof of Concept: Line 15 `string private s_password;`

Recommended Mitigation: The though or meaning behind this contract must be changed because the password can be viewed by anyone. This ruins the Reason for the creation of this contract. The `PasswordStore::s_password` is not private as its on-chain

[H-2] Missing access control, else anyone can set the password

Description: There is a missing access control allowing any user to set a password for `PasswordStore::setPassword`.The contract `PasswordStore` is inteded for the owner to be strictly responsible to providing a password.

Impact: Any user can set the password, which severly breaks the contract functioning intentions.

Proof of Concept:

▶ code

```
  function setPassword(string memory newPassword) external {
@> //@audit: no access controls
        s_password = newPassword;
        emit SetNetPassword();
    }
```

Recommended Mitigation: Add an access control. Its recommended you import Openzepplin contracts then import and inheret the ```onlyOwner`` contract

Informational

[I-1] Incorrect use of Natspec as there is no use of param as intended in the Natspec

Description: Natspect had the function `PasswordStore::getPassword` use a param `newPassword`. As displayed here the function `PasswordStore::getPassword` does not use or have any use for the param.

Impact: Natpec incorrectly used

Proof of Concept:

▶ code

```


     /*
     * @notice This allows only the owner to retrieve the password.
@>* @param newPassword The new password to set.
     */

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

```
            return s_password;
        }
```

Recommended Mitigation: Remove the Natspec Line

▶ code

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