Protocol Audit Report

Version 1.0

Cyfrin.io

November 11, 2024

Network Vulnerability Assessment

OSUOLALE

November 11, 2024

Prepared by: OSUOLALE Lead Auditors: - OSUOLALE

Table of Contents

  - [I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist causing natspec to be incorrect
  - Likelihood & Impact:

- Gas

Protocol Summary

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

Disclaimer

I, OSUOLALE 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 OSUOLALE 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 described in this document correspond the following commit hash:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

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

Roles

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

Executive Summary

The audit of the PasswordStore contract went well and was completed in two days using Foundry. Key issues included on-chain password exposure, lack of access controls on setPassword, and minor documentation errors in NatSpec comments.

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 stored on-chain is visible to anyone, and can be read directly from the blockcahin. 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

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

```
make deploy
```

3. Run the storage tool

```
We use 1 because that's the storage slot of s_password in the contract
```

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

```
0x6d79726400000000000000000000000000000000000000000000000000000008
```

4. You can the parse that hex to a string with:

```
cast aparse-bytes32-string 0
    x6d79726400000000000000000000000000000000000000000000000000000008
```

5. And get an output of:

```
myrd
```

Recommended Mitigation: Due to this, the overall architecture of the contract sould be rethought. One could encrypt the password off-chain, and then store the encrypted password on-chain. However, you'd also likely want to remove the view function as you wouldn't want the user to accidentaly send a transaction with the password that decrypts your password

Likelihood & Impact:

- Impact: HIGH
- Likelihood: HIGH
- Severity: HIGH

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

Description: The `PasswordStore::setPassword` function is set to be `external` function, however, the natspec of the function and overall purpose of the smart contract is that `This function allows only the owner to set a new password.`

```
    function setPassword(string memory newPassword) external {
          // @audit - There are no access control
        s_password = newPassword;
         emit SetNetPassword();
    }
```

Impact: Anyone can set/change the password, severely breaking the contract intended 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.prank(randomAddress);
          string memory expectedPassword = "myNewPassword";
          passwordStore.setPassword(expectedPassword);

          vm.prank(owner);
          string memory actualPassword = passwordStore.getPassword();
          assertEq(actualPassword, expectedPassword);
    }
    ```
</details>

Recommended Mitigation: Add an access control conditional to the `
    setPassword` function.

```javascript
if(msg.sender != s_owner){
```

```
        revert PasswordStore__NotOwner();
    }
```

Likelihood & Impact:

- Impact: HIGH
- Likelihood: HIGH
- Severity: HIGH

Informational

[I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist causing natspec to be incorrect

Description:

```
/*
 * @notice This allows only onwer to retrieve the password.
 * @param newPassword The new password to set
 */
function getPassword() external view returns (string memory) {}
```

The `Password::getPassword` function signature is `getPassword()` while the natspec says it should be `getPassword(string)`.

Impact: The natspec is incorrect

Recommended Mitigation: Remove the inccorect natspec line.

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

Likelihood & Impact:

- Impact: NONE
- Likelihood: HIGH
- Severity: Informational/Gas/Non-cricts

Gas