Protocol Audit Report

Version 1.0

W3dit

January 19, 2024

Protocol Audit Report

W3dit

March 7, 2023

Prepared by: W3dit Lead Security Researcher:

- I am

SOMETHING

Table of Contents

    * [H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the password
  - Likelihood & Impact:
  - Informational
    * [I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect
  - Likelihood & Impact:
  - Gas

Protocol Summary

A smart contract applicatoin 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 YOUR_NAME_HERE 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: **

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

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

Add some additional info about how this audit went, types of things you found, etc.

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 longet private.

Description: All data stored on-chain is visble to anyone, you can be read directly from the blockchain. The `PasswordStore::s_password` variable is intented to be a private varibale 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 bellow.

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

Proof of Concept:

The bellow test show 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
```

You'll get an output that looks like this:

`0x6d7950617373776f726400000000000000000000000000000000000000000014`

You can then parse that hex to a string with:

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

And get an 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.

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

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

Impact: Anyone can set/change a new password of the contract, severly 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.assume(randomAddress != owner);
            vm.startPrank(randomAddress);
            string memory expectedPassword = "myNewPassword";
            passwordStore.setPassword(expectedPassword);

            vm.startPrank(owner);
            string memory actualPassword = passwordStore.getPassword();
            assertEq(actualPassword, expectedPassword);
        }

```

```
    ```

</details>

Recommended Mitigation: Add an access controll 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 the natspec to be incorrect

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()` which is natspec say it should be `getPassword(string)`

Impact: The natspec is incorrect

Recommended Mitigation: Remove the incorrect natspec line.

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

Likelihood & Impact:

- Impact: NONE

- Likelihood: NONE
- Severity: Informational/Gas/Non-crits

Gas