Protocol Audit Report

Lydia Gyamfi Ahenkorah

November 7, 2024

Figure 1: Logo

Protocol Audit Report

Version 1.0

Bug Pirate

November 7, 2024

Prepared by: [FOAH]
Lead Auditors: Ahenkorah Gyamfi Lydia # Table of

Contents - Protocol Audit Report - Version 1.0 - Bug Pirate - Table of - Protocol Summary - Disclaimer - Risk Classification - Audit Details - Scope - Roles - Executive Summary - Issues found - Findings - High - [H-1] Variables stored in the storage on-chain are visible to anyone, no mattter the solidity visibility keyword, meaning the password is not actualy a private password. - Likelihood & Impact - [H-2] `PasswordStore::setPassword` has no access control, meaning a non owner can change the password. - Likelihood & Impact: - Informational - [I-1] The `PasswordStore::getPassword` natspec indicates parameter that doesnt exist, causing the naspec to be incorrect - Likelihood & Impact:

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 Bug Pirate 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 described in this document correspond the following commit hash

commit Hash:

`2e8f81e263b3a9d18fab4fb5c46805ffc10a9990`

Scope

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

Roles

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

Executive Summary

**I Spent 2 hous 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] Variables stored in the storage on-chain are visible to anyone, no mattter the solidity visibility keyword, meaning the password is not actualy a private password.

Likelihood & Impact

- Impact: HIGH
- Likeihood: HIGH
- Severity: HIGH

Description: All data stored on-chain is visible to anyone, and can be read directly from the blockchain. The `PasswordStore::s_password` viriable is intended to be a private viriable and only accessed through `PasswordStore::getPassword` function, which is intended to be only called by 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 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

```
cast storage 0x5FbDB2315678afecb367f032d93F642f64180aa3 1 --rpc-url http://127.0.0.1:8545
```

you'll get an output like this

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

You can then parse that hex to a string with

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

And get an output of:

```
mypassword
```

Recommended Mitigation: The whole architecture of this contract should be rethough. user should encrypt password offchain and use the encrpted password on-chain. This will require another password off-Chain the user has to remmembber to decrypt the password. Youd also likely want to remove the view function as you wouldnt want the user to mistakenly send a transaction with the password that decrypts the password.

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

Likelihood & Impact:

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

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 the `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 and change the password, severly breaking the contract 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.prank(randomAddress);
        string memory expectedPassword = "myNewPassword";
        passwordStore.setPassword(expectedPassword);

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

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

```
if(msg.sender != s_owner){
    revert PasswordStore__NotOwner();
}
```

Informational

[I-1] The `PasswordStore::getPassword` natspec indicates parameter that doesnt exist, causing the naspec to be incorrect

Likelihood & Impact:

- Impact: NONE
- Likelihood: NONE
- Severity: Informational/Gas/Non-Crits

iNFORMATIONAL: Hey, This isnt a bug but you should know...

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()` while the natspec says it should be `getPassword(string)`.

Impact: The natspec is incorect

Recommended Mitigation: Remove the incorrect natspec line

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