Protocol Audit Report

Version 1.0

Anthony E

February 10, 2024

Protocol Audit Report

Anthony E.

Feb 9, 2024

Prepared by: Anthony Lead Security Researcher: - Anthony E

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
  - Scope
  - Roles
- Executive Summary
  - Issues found
  - Findings
  - High
    * [H-1] Storing the password on-chain makes it visible to anyone, and no longer private
    * [H-2] `PasswordStore::setPassword` has no access controls, meaning non owner could change PW
      · Likelihood & Impact
  - Informational
    * [I-1] The `PasswordStore::getPassword` natspec indicates a parameter that does not exist, causing the natespec to be incorrect

Protocol Summary

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

The Anthony E 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

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 password

Executive Summary

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 blockchain. The `PasswordStore::s_password`variable is intended to be a private variable and only accesed through the `PasswordStore::getPassword` function which is intended to only be called by the owner

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

Proof of Concept: (Proof of Code)

The below testcase shows how anyone can read the PW directly from the blockchain

create local running chain

```
make anvil
```

deploy the contract to the chain

```
make deploy
```

run storage tool

use `1` beacause thats the storage slot of `s_password` in the contract we get `0x6d79506173737776f726400000000`

Recommended Mitigation: Please rethink this whole contract as the encryption of passwords are null and void if you choose to store the pw on the blockchain

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

Description: The `PasswordStore::setPassword` function is set to be an `external` function however, this brings the overall function of the smart contract contract invalid `This function omly allows owner to set new PW`

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

Impact: Anyone can set/change the 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.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 conditional to the `setPassword`function

```
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 does not exist, causing the natespec to be incorrect

Impact: The natspec is incorrect

Recommended Mitigation: Remove incorrect natspec line

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