



Protocol Audit Report

Prepared by: Heng-Syu Lin ()

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*\(Root cause\) *makes it visable to anyone and no longer private*\(Impact\).](#)
 - [Likelihood & Impact:](#)
 - [\[H-2\] `PasswordStore::setPassword\(\)` has no access controll, 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:](#)

Protocol Summary

PasswordStore is a protocol dedicated to 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 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

Impact				
	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 the password and read the password.
- Outsiders: No one else should be able to set or read the password.

Executive Summary

Add some notes about how the audit went, types of things you found, etc. We spent X hours with Z auditors using Y tools. etc*

Issues found

Severity	Nubers of issues found
High	2
Medium	0
Low	0
Info	1
Total	3

Findings

High

[H-1] *Storing the password on-chain*(Root cause) *makes it visible to anyone and no longer private*(Impact).

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

We show one such method of reading any data off chain below

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 local running chain

```
make anvil
```

2. Deploy 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 and output that looks like this:

```
0x6d79506173737766726400000000000000000000000000000000000000000014
```

You can then parse that hex to a string with:

```
cast parse-bytes32-string  
0x6d795061737377667264000000000000000000000000000000000000000014
```

And get the output:

```
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 (There's a severe disruption of protocol functionality)
- Likelihood : High (It's highly probable to happen)
- Severity : High
- !CRIT

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

Description: The `PasswordStore::setPassword` function is set to be an `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 is no access controll
        s_password = newPassword;
        emit SetNetPassword();
}
```

Impact: Anyone can set/change the password of the contract, severely breaking the contract intended functionality.

Proof of Concept: Add the following code to `PasswordStore.t.sol` test file:

► Code

```
function test_anyone_can_change_password(address randomAddress) public {
    vm.assume(randomAddress != owner);
    vm.prank(randomAddress);
    string memory newPassword = "YouBeenHacked";
    passwordStore.setPassword(newPassword);

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

Recommended Mitigation: Add an access control conditional to the `PasswordStore::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 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) {
    if (msg.sender != s_owner) {
        revert PasswordStore__NotOwner();
    }
    return s_password;
}
```

The `PasswordStore::getPassword` function signature is `getPassword()` while the natspec says 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