



Protocol Audit Report

Version 1.0

Cyfrin.io

February 11, 2025

Protocol Audit Report

M.forte

February 11th 2025

Prepared by: m.forte Lead Auditors: - xxxxxxxx

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
 - Scope
 - Roles
- Executive Summary
 - Issues found
- Findings
- High
 - [H-1] Storing password on-chain makes it visible to anyone, and no longer private
 - [H-2] `PasswordStore::setPassword` Missing Access Control. i.e non-owner could change password
- Informational
 - [I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect.

Protocol Summary

PasswordStore is a protocol dedicated to storage and reteieval of a user's passwords. The protocol is designed to be used by a single user, and 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
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 in this document correspond with the following commit hash: [7d55682ddc4301a7b13ae94130](#)

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	Number of issues found
High	2
Medium	0
Low	0
Info	1
----	-----
Total	3

Findings

High

[H-1] Storing 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 blockchain. 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 by the contract owner.

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 locally running chain

```
1 make anvil
```

2. Deploy the contract to the chain

```
1 make deploy
```

3. Run the storage tool We use 1 because that's the storage slot of `s_password` in the contract.

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

You'll get an output that looks this: `0x6d7950617373776f726400`

You can then parse the hex to string with:

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

And get an output of:

```
1 myPasword
```

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 functions as you wouldn't want the user to accidentally send a transaction with the password that decrypts your password.

[H-2] PasswordStore::setPasswordMissing Access Control. i.e non-owner could change password

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

Code

```
1 function setPassword(string memory newPassword) external {
2     // @ audit - There are no access controls
3     s_password = newPassword;
4     emit SetNetPassword();
5 }
```

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

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

Code

```
1 function test_anyone_can_set_password( address randomAddress) public
2 {
3     vm.assume(randomAddress != owner);
4     vm.prank(randomAddress);
5     string memory expectedPassword = "myNewPassword";
6     passwordStore.setPassword(expectedPassword);
7
8     vm.prank(owner);
9     string memory actualPassword = passwordStore.getPassword();
10    assertEq(actualPassword, expectedPassword);
11 }
```

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

```
1 if(msg.sender != s_owner) {
2     revert PasswordStore__NotOwner();
3 }
```

Informational

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

Description:

```
1 /*
2  * @notice This allows only the owner to retrieve the password.
3  * @param newPassword The new password to set.
4  */
5 function getPassword() external view returns (string memory) {
```

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

Impact: The natspec is incorrect

Recommended Mitigation: Remove the incorrect natspec line.

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