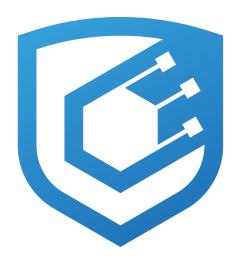
Protocol Audit Report Cyfrin.io March 7, 2023



Protocol Audit Report

Version 1.0

Cy frin.io

Protocol Audit Report

Cyfrin.io

March 7, 2023

Prepared by: Krunal Lead Auditors: - Krunal Savaliya

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.
 - Likelihood & Impact:
 - * [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:

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		
Likelihood	High Medium Low	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 the document correspond the following commit hash:

2e8f81e263b3a9d18fab4fb5c46805ffc10a9990

Scope

./src/

|- PasswordStore.sol

Roles

- Owners: 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 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 accessed through 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 below.

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

Proof of Concept: (Proof of Code)

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

4. You'll get an output that looks like this:

5. You can then parse that hex to a string with:

6. 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 stored password. However, you're also likely want to remove the view function as you wouldn't want the user to accidentally send a transaction with this decryption key.

My Recommendation: The password could be encrypted off-chain and then store the encrypted password on-chain. User would have to remember another password off-chain to decrypt the stored password. Remove the view function as you wouldn't want the user to accidentally send a txn with this decryption key.

Likelihood & Impact:

Impact: HIGHLikelihood: HIGH

• Severity: HIGH (or CRITICAL)

[H-2] PasswordStore::setPassword has no access controls, 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 are no access controls
s_password = newPassword;
emit SetNetPassword();
}
```

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

Proof of Concept: Add the following to the PasswordStore.t.sol 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: HIGHLikelihood: HIGHSeverity: 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.

// @audit - There is no newPassword parameter!

* @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 incorrect.

Proof of Concept:

Recommended Mitigation: Remove the incorrect natspec line.

* @param newPassword The new password to set.

Likelihood & Impact:

Impact: NONELikelihood: NONE

• Severity: Informational/Gas/Non-crits