Protocol Audit Report

Zach Katz

April 22, 2025

Contents

PasswordStore Audit Report

Version 1.0

ZKAudits

April 22, 2025

Protocol Audit Report

Zach Katz

April 22, 2025

Prepared by: [Zach Katz] Lead Auditors: - Zach Katz

Table of Contents

Protocol Summary

The protocol attempts to allow only the owner of the smart contract to set and get a password.

Disclaimer

The DiscoAudit 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 to the following commit hash:

`7d55682ddc4301a7b13ae9413095feffd9924566`

Scope

- In Scope:

```
./src/
  PasswordStore.sol
```

Roles

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

Executive Summary

Severity	Number of issues found
Highs	2
Medium	0
Low	0
Info	1
Total	3

Issues found

Findings

High

[H-1] Password stored on-chain is visible to anyone and not private.

Description: All data stored on-chain is visible to anyone, and can be read directly from the blockchain. The `PasswordStore::s_password` variable s intended to be a private variable and only accessed through through the `PasswordStore::getPassword` function, which is inteded 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, severely breaking the functionality of the protocol.

Proof of Concept:

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 <ADDRESS_HERE> 1
```

You can then parse that hex to a string with:

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

And get an output of:

```
myPassword
```

Recommended Mitigation: The architecture of the contract should be rethought. The password could be encrypted off-chain, and then the encrypted version stored on-chain.

Likelihood & Impact:

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

[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 cntract is that `This function allows only the owner to set a new password.` The function does not check if the setter is the contract owner, and thus anyone can change the 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, severely breaking the intended contract functionality.

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

```
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 `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` natspac specificies a parameter that doesn't exist

Description:

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: N/A
- Severity: Informational