Protocol Audit Report

Version 1.0

Miriam Shaka

April 22, 2025

Protocol Audit Report

Miriam Shaka

April 22, 2025

Table of Contents

Protocol Summary

Protocol does X, Y, Z

Disclaimer

All effort has been made 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

A smart contract applicatoin for storing a password. Users should be able to store a password and then retrieve it later. Others should not be able to access the password.

Scope

- Commit Hash: 2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
- In Scope:

```
./src/
#-- PasswordStore.sol
```

- Solc Version: 0.8.18
- Chain(s) to deploy contract to: Ethereum

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

Issues found

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

Findings

High

[H-1] Storing the password on-chain makes it viasable to anyone, and no longer private

Description All data stored on-chain is visible to anyone, and can be read directly from the cblockchain. The `PasswordStore::getPassword` function, which is intended to be only called by the owner of the contract.

We shoe one such method of reading any data off cahin below.

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

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

```
make anvil
```

2. Deploy the contract to the chain

```
make deploy
```

3. Run the storage tool

```
cast parse-bytes32-string 0x6d7950
```

And get an output of:

```
myPassword
```

Recommended Mitigations: Due to this, the overall architecture of the contract should be rethought. One could encrypt the password off-cahin and the store the encrypted password on-chain. This would require the user to remember another password off-chain to decrypt the password. However, you'd a;so likely want to remove the view function as you wouldn't want the user to accidentally send a transaction with the password that decrypt your password.

[H-2] `PasswordStore::setPassword` has no acess control, 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.`

Code

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

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

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

Code

```
function test_anyone_can_st_password(address randomAddress) public {
    vm.assume(randomAddress);
    vm.prank(randomAddress);
```

```
4       string memory expectedPassword = "myPassword";
5
6       vm.prank(randomAddress);
7       string memory actualPassword = password.getPassword();
8       assertEq(actualPassword, expectedPassword);
9   }
```

Recommended Mitigations: Add an access control condition to the `setPassword` function.

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

Informational

[I-1] The `PasswordStore::getPassword` nutspec indicates a parameter that doesnt exist, causing the nutspec 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       //no param newPassword passed probably documentation error
```

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

Impact: The nutspec is incorrect

Recommended Mitigations: Remove the incorrect nutspec line

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