



# **Protocol Audit Report**

Version 1.0

*<https://github.com/Patrick-Ngururi>*

June 26, 2025

# Protocol Audit Report

Patrick Ngururi

June 26, 2025

Prepared by: Patrick Ngururi 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 the password on-chain makes it visible to anyone and no longer private
    - \* [H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the 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 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 Patrick’s 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 descrobed in this document correspond to the following commit hash:

1 2e8f81e263b3a9d18fab4fb5c46805ffc10a9990

Scope

```
1 ./src/  
2 #-- 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

### Issues found

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

## 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 public and visible to anyone. The `PasswordStore::s_password` variable is intended to be hidden and only accessible by the owner through the `PasswordStore::getPassword` function.

I show one such method of reading any data off chain below.

**Impact:** Anyone is able to read the private password, severely breaking the functionality of the protocol.

**Proof of Concept:** The below test case shows how anyone could read the password directly from the blockchain. We use foundry's cast tool to read directly from the storage of the contract, without being the owner.

## Create a locally running chain

```
1 make anvil
```

## Deploy the contract to the chain

```
1 make deploy
```

## 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 like this:

[illegible]

You can then parse that hex to a string with:

```
1 cast parse-bytes32-string 0  
x6d792050617373776f726400000000000000000000000000000000000000000000000000000000
```

And get an output of:

```
1 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.

**[H-2] PasswordStore::setPassword has no access controls, meaning a non-owner could change the password**

## Informational

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