# Smart Contract Audit Report

**Auditor:** 0xGutzzz
**Date:** 07-09-2025
**Project Name:** [Password Protocol]
**Scope:** [src/PasswordStore.sol]

---

## Table of Contents



---

## Introduction

This contract allows you to store a private password that others won't be able to see. You can update your password at any time.

---

## Executive Summary

- **Overall Risk Level:** [Low / Medium / High / Critical]
- **Total Findings:** 2
  - Critical: 2
  - High: 0
  - Medium: 0
  - Low: 0

- Informational: 0

Summarize key findings and overall project security posture.

---

## Audit Scope

- Files reviewed:
  - `PasswordStore.sol`
- Lines of code: 43

---

## Methodology

Explain how the audit was performed: - Manual review

---

## Findings Summary

| ID | Severity | Title | Status |
|---|---|---|---|
| F-01 | Critical | Private Information Disclosure | Practice Audit |
| F-02 | Critical | Missing Access Control | Practice Audit |

---

## Detailed Findings

### Finding F-01: [Private Information Disclosure]

- **Severity:** Critical

- **Status:** Practice Audit

- **Description:**
  s_password is stored as a private variable. This only restricts access from other contracts and functions within the code—it does not prevent anyone from reading the variable's value directly from the blockchain. All contract storage is publicly accessible, even if marked private, because blockchain data is transparent by design, making the user's password visible to anyone on-chain

```
//@ audit -> The password is stored in plain text, making it easily accessible to any
    string private s_password;
```

#### Finding F-02: [Missing Access Control in `PasswordStore::setPassword` Allowing Bad Actor To Change User's Password]

- **Severity:** Critical

- **Status:** Practice Audit

- **Description:**
  `PasswordStore::setPassword` allows any external account to change the stored password by calling the function and providing a new password string. The function is marked as external, which means it can be invoked by any address, including users and other contracts.

```
//@ audit -> Missing access control on setPassword function, allowing anyone to chan
    function setPassword(string memory newPassword) external {
        s_password = newPassword;
        emit SetNewPassword();
    }
```

### Recommendations

#### Finding F-01: [Private Information Disclosure]

**Recommendation** Consider storing only password hashes or using off-chain solutions for sensitive data.

### Finding F-02: [Missing Access Control in `PasswordStore::setPassword` Allowing Bad Actor To Change User's Password]

**Recommendation** To fix the missing access control vulnerability, add an owner check to the setPassword function `solidity    //@ audit -> Added access control to restrict password changes to the owner only.     function setPassword(string memory newPassword) external {         if (msg.sender != s_owner) {         revert PasswordStore__NotOwner();         }`