# Protocol Audit Report

Version 1.0

*Abyss*

August 1, 2025

# Protocol Audit Report

Abyss

August 1, 2025

Prepared by: Abyss Lead Auditors: - Abyss

## Table of Contents

## 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 Abyss 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

Commit Hash:

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

### Scope

```
./src/
#-- 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

*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 visable 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 private variable and only accessed through the `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:** Any can read the private password, severly breaking the functionality of the protocol.

**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

We use 1 because that´s the storage slot of `PasswordStore::s_password` in the contract.

```
cast storage <ADDRESS_HERE> 1 --rpc-url 127.0.0.1:8545
```

You´ll get an output that looks like this:

`0x6d7950617373776f726400000000000000000000000000000000000000000014`

You can then parse that hex to a string with:

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

And get an output of:

```
myPassword
```

**Recommended Mitigation:**

Do **not** store or transmit plaintext passwords on-chain, even as function parameters, because all transaction data is public. Instead:

- Store only a hash of the password off-chain, and never send the plaintext password to the contract.
- If on-chain verification is required, use a challenge-response protocol or zero-knowledge proofs so the password itself is never sent or stored on-chain.
- For most use cases, keep all sensitive authentication and password management off-chain.

**Example (off-chain verification):** - Store a hash of the password off-chain. - When authentication is needed, perform the check off-chain and only interact with the contract for non-sensitive actions.

**Example (hashing, but still not fully secure):**

```
// This still exposes the password as a parameter!
function setPassword(string memory newPassword) external onlyOwner {
    s_passwordHash = keccak256(abi.encodePacked(newPassword));
}
```

**Warning:** Even with hashing, the password is visible in the transaction input data. Do not use this pattern for real secrets.

**Summary:**
Never send or store sensitive data (like passwords) on-chain or as transaction parameters. Use off-chain authentication or advanced cryptographic techniques (e.g., zero-knowledge proofs) if on-chain verification is absolutely required.

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

**Description:** The password `PasswordStore::setPassword` function is set to ba 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 controlls
        s_password = newPassword;
        emit SetNetPassword();
    }
```

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

**Proof of Concept:** Add the following to the `PasswordStore.t.sol` test 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();
}
```

## 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.
@>  * @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.

**Recommended Mitigation:** Remove the incorrect natspec line.

```
/*
 *@notice This allows only the owner to retrieve the password.
-*@param newPassword The new password to set.
*/
```