Protocol Audit Report

Version 1.0

Cyfrin.io

September 15, 2025

Protocol Audit Report

YoYiL

September 15, 2025

Prepared by: YoYiL Lead Auditors:

- YoYiL

Table of Contents

Protocol Summary

PasswordStore is a protocol for storing a password. The protocol is designed to be used by a single user. Users should be able to store a password and then retrieve it later. Others should not be able to access the password.

Disclaimer

The YoYiL 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 described in this document correspond the following 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

This audit (scope: single file PasswordStore.sol, commit 7d55682ddc4301a7b13ae9413095feffd9924566) identified 4 issues: 2 High, 1 Low, 1 Informational (no Medium or Gas). The two High issues fundamentally break the intended guarantees: (1) the password is stored in plaintext on-chain and is trivially readable by anyone, so privacy is impossible; (2) setPassword lacks access control, allowing any address to overwrite the value, destroying integrity and enabling denial of service. As designed, the contract cannot fulfill a “private storage” claim. Recommended actions: redesign to use off-chain encryption with only a ciphertext or hash on-chain, implement strict ownership checks, avoid returning plaintext (or clearly disclaim the absence of secrecy), and fix misleading NatSpec before seeking re-audit.

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.

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 `s_password` in the contract.

```
cast storage <ADDRESS_HERE> 1 --rpc-url http://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: 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.

Description:

```
1 /*
2    * @notice This allows only the owner to retrieve the password.
3 @> * @param newPassword The new password to set.
4 */
5 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.

```
1      /*
2 +     * @notice This allows only the owner to retrieve the password.
3 -     * @param newPassword The new password to set.
4       */
```