DakSec
SECURITY

Password Store Audit Report

Version 1.0

cyfrin.io

June 30, 2025

Protocol Audit Report

Dakota Rogers

June 5, 2025

Prepared by: [DakSec Security] Lead Auditors: - Dakota Rogers

Table of Contents

Protocol Summary

The `PasswordStore` contract is designed to store and retrieve a password string on-chain. The core functionalities include:

Allowing the contract deployer (owner) to set and retrieve a password.

Restricting password visibility to the contract owner.

Emitting an event when the password is updated.

However, the protocol makes critical assumptions about data privacy on the blockchain — namely, that private variables cannot be observed externally. In practice, all on-chain data is publicly accessible via blockchain nodes and explorers, even if marked **`private`** in Solidity. This results in the core functionality (secure password storage) being fundamentally insecure by design.

Disclaimer

The DakSec Security 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
	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

The findings described in detail in this document correspond the following commit hash:

Commit Hash: 7d55682ddc4301a7b13ae9413095feffd9924566

Scope

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

Roles

Executive Summary

The DakSec Security team conducted a security audit of the PasswordStore smart contract. The audit focused on identifying common vulnerabilities, logic flaws, and architectural misassumptions. The audit revealed two critical issues and one informational finding. Notably, the contract assumes sensitive data (password) can be hidden on-chain, which is a fundamental misunderstanding of the Ethereum data model.

Audit Duration: 2 hours

Auditor(s): 1

Tools Used: Manual review, Foundry test framework, VS Code, Etherscan

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 visible 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 a private variable and only accessed through the `PasswordStore::getPassword` function, which is intended to be only called by the owner of the contract.

Impact: Anyone 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.

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 password. However, you'd also likley want to remove the view function as you wouldn't want the user to accidentally send a transaction with the password that decrypts your password.

[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 exsist, 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.

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