

PasswordStore Audit Report

Sidrah Ahmed

January 10, 2024

PasswordStore Initial Audit Report

Version 0.1

January 10, 2024

PasswordStore Audit Report

Sidrah Ahmed

January 10, 2024

PasswordStore Audit Report

Prepared by: Sidrah Ahmed

Auditor:

- Sidrah Ahmed <https://github.com/sidrahh12>

Assisting Auditors:

- None

Table of Contents

See Table

- PasswordStore Audit Report
- Table of Contents
- About the Auditor
- Disclaimer
- Risk Classification
- Audit Details
- Scope
- Protocol Summary
- Roles
- Executive Summary
- Issues Found
- Findings
- High
- [H-1] Passwords stored on-chain are visible to anyone, regardless of solidity variable visibility
- [H-2] `PasswordStore::setPassword` is callable by anyone
- Low Risk Findings
- L-01. Initialization Timeframe Vulnerability
- Relevant GitHub Links
- Summary
- Vulnerability Details

- Impact
- Tools Used
- Recommendations
- [I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect

About the Auditor

Disclaimer

I make all efforts to find as many vulnerabilities in the code within the given time period, but hold no responsibilities for the findings provided in this document. A security audit by me 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		
Likelihood	High	High	Medium	Low
	Medium	H/M	M	M/L
	Low	M	M/L	L

Audit Details

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

2e8f81e263b3a9d18fab4fb5c46805ffc10a9990

Scope

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

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.

Roles

- Owner: Is the only one who should be able to set and access the password.

For this contract, only the owner should be able to interact with the contract.

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] Passwords stored on-chain are visible to anyone, not matter solidity variable visibility

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.

However, anyone can directly read this using any number of off chain methodologies

Impact: The password is not private.

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


```
if (msg.sender != s_owner) {  
    revert PasswordStore__NotOwner();  
}
```

Low Risk Findings

L-01. Initialization Timeframe Vulnerability

Submitted by dianivanov.

Relevant GitHub Links

<https://github.com/Cyfrin/2023-10-PasswordStore/blob/main/src/PasswordStore.sol>

Summary

The PasswordStore contract exhibits an initialization timeframe vulnerability. This means that there is a period between contract deployment and the explicit call to setPassword during which the password remains in its default state. It's essential to note that even after addressing this issue, the password's public visibility on the blockchain cannot be entirely mitigated, as blockchain data is inherently public as already stated in the "Storing password in blockchain" vulnerability.

Vulnerability Details

The contract does not set the password during its construction (in the constructor). As a result, when the contract is initially deployed, the password remains uninitialized, taking on the default value for a string, which is an empty string.

During this initialization timeframe, the contract's password is effectively empty and can be considered a security gap.

Impact

The impact of this vulnerability is that during the initialization timeframe, the contract's password is left empty, potentially exposing the contract to unauthorized access or unintended behavior.

Tools Used

No tools used. It was discovered through manual inspection of the contract.

Recommendations

To mitigate the initialization timeframe vulnerability, consider setting a password value during the contract's deployment (in the constructor). This initial value

can be passed in the constructor parameters.

[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 natspec for the function PasswordStore::getPassword indicates it should have a parameter with the signature getPassword(string). However, the actual function signature is getPassword().

Impact: The natspec is incorrect.

Recommended Mitigation: Remove the incorrect natspec line.

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