PasswordStore Audit Report

Version 1.0

056.xyz

January 12, 2025

PasswordStore Audit Report

056xyz

Jan 5, 2025

Prepared by: [056xyz] Security Researcher: - 056xyz

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
  - Scope
    * –src
  - Roles
- Executive Summary
  - Issues found
- Findings
  - High
    * [H-1] Storing the password on-chain makes it visible to anyone
    * [H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the password
  - Info
    * [I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect.

Protocol Summary

Protocol to store password on-chain

Disclaimer

The 056xyz 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

Scope

–src

Roles

Executive Summary

Issues found

Severity	N of Issues
High	2
Medium	
Low	
Info	1
Total	3

Findings

High

[H-1] Storing the password on-chain makes it visible to anyone

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. **Impact:** Anyone is able to read the private password, severly 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.

```
Create a locally running chain
```

make anvil

```
Deploy the contract to the chain
```

make deploy

```
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 0x6d7950617373776f7264000000000000000000000000000000000000000000000014
```

And get an output of:

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

Description: `The PasswordStore::setPassword` function is set to be an external function, however the purpose of the smart contract and function's natspec indicate that This function allows only the owner to set a new password.

Impact: Anyone can set/change the stored password, severly breaking the contract's intended functionality

Proof of Concept:

```
function test_anyone_can_set_password(address randomAddress) public {
        vm.assume(randomAddress != owner);
        vm.startPrank(randomAddress);
        string memory expectedPassword = "myNewPassword";
        passwordStore.setPassword(expectedPassword);

        vm.startPrank(owner);
        string memory actualPassword = passwordStore.getPassword();
        assertEq(actualPassword, expectedPassword);
    }
```

Recommended Mitigation: Add an access control conditional to `PasswordStore::setPassword`
.

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

Info

[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:

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