PasswordStore Audit Report

Version 1.0

Izuman

February 1, 2024

Practice Audit Report

IzuMan

February 1, 2024

Prepared by: IzuMan

Lead Security Expert: IzuMan

Table of Contents

Protocol Summary

PasswordStore is a protocol that allows only the designated owner to store and retrieve passwords in the contract storage.

Disclaimer

IzuMan 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 this dcoument correspond to the following commit hash:

```
aiosdjasdfijimmii090n34naklnamnnxdajhuiq4
```

Scope

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

Roles

- Owner: The user who can set and retrieve the password
- Outsider: All other addresses should not be able to set or retrieve the password

Executive Summary

Major issue were found within the smart contract and should not be interacted with until they are fixed

Issues found

Severity	Number of Issues found
High	2
Medium	0
Low	0
Info	1
Total	3

Findings

High

[H-1] The password variable stored on-chain storage is visible to anyone

Description: All data stored on-chainis visible to anyone, and can be read directly from the blockchain. The `PasswordStore::s_password` variable is intended to be private for only the owner of the contract.

We show one such emthod of reading any off chain below

Impact Anyone can read the private password, severely breaking the functionality of the protocol

Proof of Concept: (Proof of Code) Below shows how anyone can read contract storage off the blockchain

1

```
    make anvil
```

2. Deploy the contract to the chain. `make deploy`
3. Run the storage tool We use `1` because that is the storage slot of `PasswordStore::s_password` in the contract

```
cast storage <CONTRACT_ADDRESS> --rpc-url http://127.0.0.1:8545
```

it will return bytes32 data:

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

now convert the bytes32 data to a string

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

This will give us an output of

```
myPassword
```

Recommended Mitigation: This is an architectual error storing non-encrypted data on-chain. One solution is to encrypt the password off chain with a secret key, then put the encrypted key on-chain. Also, the getPassword function should be removed to prevent accidently displaying your secret key. However, this solution will make the owner store a secret key off-chain.

Likelihood and Impact:

- Impact: High
- likelihood: High
- Severity: High

[H-2] `PasswordStore::setPassword` has no acces controls which mean anyone can change the password

Descripion: Anyone can call the function `PasswordStore::setPassword` with a string which will be set as the new password. Also, the natspec of the contract says `This allows only the owner to retrieve the password.`

```
function setPassword(string memory newPassword) external {
        s_password = newPassword;
```

```
3 @>      // @audit - there are no access controls
4         emit SetNetPassword();
5     }
```

Impact Anyone can change the password of the contract

Proof of Concept: Add the following to `PasswordStore.t.sol` test file.

code

```
1
2 function test_non_owner_reading_password_reverts() public {
3         vm.startPrank(address(1));
4         vm.expectRevert(PasswordStore.PasswordStore__NotOwner.selector)
              ;
5         passwordStore.getPassword();
6 }
```

Recommended Mitigation: Add an access control like the following

```
1 if(msg.sender != s_owner){
2     revert PassWordStore__NotOwner();
3 }
```

Likelihood and Impact:

- Impact: High
- likelihood: High
- Severity: High

Informational

[I-1] The `PasswordStore::getPassword` natspec indicates there shoould be a parameter that doesn't exist, natspec is incorrect

Descripion: From the natspec documentation `@param newPassword The new password to set.` However, there is no param newPassord that exists in the function.

Impact The natspec is incorrect

Recommended Mitigation:

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

Likelihood and Impact:

- Impact: None
- likelihood: Low
- Severity: Informational /Gas/Non-crits