PasswordStore Protocol Audit Report

Version 1.0

TrustWarden

July 25, 2024

PasswordStore Protocol Audit Report

TrustWarden

July 25, 2024

Prepared by: TrustWarden

Lead Security researcher:

- TrustWarden

Table of Contents

Protocol Summary

PasswordStore Protocol gives the ability to one user (the owner) to store and retreive a password onchain, it is a protocol desined for using single user and not be used by multiple users, meaning but the owner, the others can't save new password or retrive the last one.

Disclaimer

The TrustWarden 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 this document correspond the following commit hash:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

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

We spent 4hrs approximately in learning process to get familiar with security researching and auditing smart contracts. It was only me that conduct this auditing and watching from Cyfrin course. We didn't use any specific tools just manual reviwing for this auditing, we found 2 -HIGH- severities in the code base that severly breaking the functionality of the entire protocol's purpose, and also found one Informational issue in the natspec of the `PasswordStore::getPassword` function.

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.

We show one such method of reading any data off chain below.

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.

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 <CONTRACT_ADDRESS_HERE> 1 --rpc-url http://127.0.0.1:8545
```

You'll get an output that looks like this: `0x6d7950617373776f726400000000000000000000000000000000000`

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. the current architecture implemented of the contract is to keep the `PasswrodStore::s_password` private, and others can't accessed and read it, and onbviously there isn't a private data on the blockchain. recommend to encrypt the password off-chain and then call the `PasswordStore::setPassword` function with the encrypted data to prevent non-owner users can understand what the real password is, but with this method the owner need to remember what the password is in the first place before encrypt it.

[H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the password

Description: The `PasswordStore::setPassword` is intended to be called by only the owner, but there isn't any checks to validate the caller, so anyone could call the `PasswordStore::setPassword` function and change the password at any time. They: The `PasswordStore::setPassword` is set to be an `external` function, however, the natspec of the function and overall purpose of the smart contract is that `This function allows only the owner to set a new password.`

```
1      function setPassword(string memory newPassword) external {
2  @>      // @audit-> there are no access control
3          s_password = newPassword;
4          emit SetNetPassword();
5      }
```

Impact: Anyone can set/change `PasswordStore::s_password` of the contract, severly breaking the contract intended functionality.

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

Code

```
1      function test_anyone_can_set_password(address randomAddress) public
           {
2          vm.assume(randomAddress != owner);
3          vm.prank(randomAddress);
4          string memory expectedPassword = "myNewPassword";
5          passwordStore.setPassword(expectedPassword);
6
7          vm.prank(owner);
8          string memory actualPassword = passwordStore.getPassword();
9          assertEq(actualPassword, expectedPassword);
10     }
```

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

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

Informational

[I-1] The `PasswordStore::setPassword` natspec indicates a parameter that doesn't exist, causing the natspec 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()` which the natspec says it should be `getPassword(string)`.

Impact: The natspec/documentation is incorrect.

Recommended Mitigation: Remove the incorrect natspec line.

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