Protocol Audit Report

Version 1.0

hoBabu

November 3, 2024

Protocol Audit Report

hoBabu

November 3, 2024

Prepared by: [hoBabu]

Table of Contents

Protocol Summary

A smart contract applicatoin for storing a password. Users should be able to store a password and then retrieve it later. Others should not be able to access the password.

Disclaimer

The YOUR_NAME_HERE 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

Scope

Roles

Executive Summary

Issues found

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

Findings

High

[H-1] Password store in private variable is not private, its just limited to solidity. Password is visible to anyone and no longer private.

Description: All data that is stored on-chain is visible to everyone, `PasswordStore::s_password` which stores the password of the user is supposed to be private and can be only accessed by using `PasswordStore::getPassword()`, this password will return the password if its called by owner.

Impact: Anyone can read the password of user, severly breaking the functionality of the protocol.

Proof of Concept: Proof of Code 1. Start a Local chain

```
make anvil
```

2. Deploy the contract

```
make deploy
```

3. Read the storage variable

```
cast storage 0x5FbDB2315678afecb367f032d93F642f64180aa3 1 --rpc-url
    http://127.0.0.1:8545
```

4. Convert bytes into String

```
cast parse bytes32-string 0
    x694c6f7665596f75000000000000000000000000000000000000000000000010
```

5. Password retrived

```
iLoveYou
```

Recommended Mitigation: Due to this overall architecture of the contract should be rethought. One could encrypt the password off-chain, and then store the encrypted password on-chain.

[H-2] `PasswordStore::setPassword` has no access contol. Anyone can come and set the password.

Description: `PasswordStore::setPassword` function natspec states that `This function allows only the owner to set a new password.`

```
    function setPassword(string memory newPassword) external {
        // @auidt - no access control
@>          s_password = newPassword;
        emit SetNetPassword();
    }
```

Imapct Anyone can come and chanbge the password, severly breaking the functionality of the protocol.

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

Code

```
    function test_non_owner_can_setPassword( address randomAddress)
        public {
        vm.assume(randomAddress != owner);
        vm.prank(randomAddress);
        string memory newPassword = "Your protocol has been RekT";
        passwordStore.setPassword(newPassword);
        vm.prank(owner);
        string memory actualPassword = passwordStore.getPassword();
        assertEq(actualPassword,newPassword);

    }
```

Recommended Mitigation:
Access control check can be added in the function `PasswordStore::setPassword`

```
if(msg.sender != owner){
    revert("Non-Owner is trying to set the password ")
}
```

Informational

[I-1] Wrong natspec for the `PasswordStore::getPassword`, misleading information for the developers.

Description Function `PasswordStore::getPassword` have this natspec `/** @notice This allows only the owner to retrieve the password. * @param newPassword The new password to set. */` As per the natspec `newPassword` needed to be given in the parameter of function `PasswordStore::getPassword` but its of no use.

Imapct: Incorrect natspec.

Recommended Mitigation: Remove this line fronm the function natspec

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