Protocol Audit Report

Version 1.0

Keyword

January 2, 2024

Protocol Audit Report

Keyword

Jan 1st, 2024

Prepared by: Keyword

Table of Contents

Protocol Summary

Protocol does X, Y, Z

Disclaimer

Keyword 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

Roles

Executive Summary

Issues found

Findings

High

[H-1] Storing the password onchain 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 blockchain below.

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

Proof of Concept:

The below test case shows how anyone can read the password directly from the blockchain.

1. Create a local blockchain network

```
make anvil
```

2. Deploy the contract on-chain

```
forge script script/DeployPasswordStore.s.sol:DeployPasswordStore --rpc-url http://localhost:8545 --private-key 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80 --broadcast
```

3. Read from storage We use 1 because `s_password` is stored at #1 index in storage.

```
cast storage {Address here} 1 --rpc-url http://127.0.0.1:8545
```

You'll get an output like this 0x6d7950617373776f726400000000000000000000000000000000000000000014

Now parse that hex to a string with cast parse

```
cast parse-bytes32-string 0
    x6d7950617373776f726400000000000000000000000000000000000000000014 (
    you will put your own hex value here)
```

You will get an output of

```
myPassword
```

Recommended Mitigation: Due to this bug the whole architecture of the contract should be rethought.

[H-2] `PasswordStore::setPassword` has no access controls, meaning that anyone can can change the password, not only the owner.

Description: The `PasswordStore::setPassword` is set to be an external function that should be accessed only by the owner of the cotract, but the function misses an access control feature, so any non-owner could change the password.

Impact: Anyone can set/change the password of the contract, severly impacting its purpose.

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

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

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

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

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

Informational

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

Impact: The natspec is incorrect

Recommended Mitigation: Remove the incorrect natspec.

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