Protocol Audit Report

Version 1.0

Cyfrin.io

December 21, 2023

Protocol Audit Report

Atoko

March 7, 2023

Prepared by: Atoko Lead Auditors: - xxxxxxx

Table of Contents

Protocol Summary

Protocol does X, Y, Z

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

Findings

High

[H-1] Storing the password on chain makes it visible to anyone and no longer private

Desctiption: All the data that is stored on chain is visible to anyone and can be read by anyone, the `PasswordStore::s_password` is a private variable and can only be accessed through the `PasswordStore::getPassword` function which is intended to be only called by the owner of the contract.

we show such method of reading any data on chain below.

Impact: Anyone can read the private password severely breaking the functionality of the contract

below is a test case that shows that anyone is able to read the private storage

1. Create a locally running chain

```
make anvil
```

2. now that you have that deploy the contract to get the address of the contract

```
make deploy
```

3. after you get the contract address use foundrys cast to get the value of storage

```
cast storage < contract address > and the storage spot of our
    s_password in our case 1
```

4. Get the actual representation of the bytes you will get from running the above command

```
cast parse-bytes32-string <enter the bytes here to see the password>
```

Mitigation:

Due to this the overall architecture of the protocol should be rethought, ie one could encrypt the password offchain and store the encrypted password onchain this would require another user to remember the password offchain, however, you would want to remove the view function to prevent users to accidentaly send a transaction with the password that decrypts your password

[H-2] `PasswordStore::setOwner` function has no access controll meaning that a non owner can set the password

Description:

The `PasswordStore::setOwner` function has no access controll but it should only be called by the owner based on the natspec of the function `'This function allows only the owner to set a new password.'` what this means is that anyone who wants to set the password of your password store will be able to change the password since it is an external function.

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

Impact: Anyone can set/change the password of the contract severery breaking the intended functionality of the contract

Proof of concept: Add the following to passwordStore.t.sol test file and run the test

```
function testAnyOneCanSetPassword(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 Mitigaion:

Add access controll to the function so that only the owner will bne able to access the function.

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

Medium

Low

Informational

[l-1] The natspec indicates a parameter that does not exist making the natspect to be incorrect

```
  /*
     * @notice This allows only the owner to retrieve the password.
     * @param newPassword The new password to set.
     */
    function getPassword() external view returns (string memory) {
        if (msg.sender != s_owner) {
            revert PasswordStore__NotOwner();
        }
        return s_password;
    }
```

Recommended Mitigation: remove the line indicating the newPassword parameter # Gas