Smart Contract Audit Report

General Information

- **Project Name:** Password Store
- **Smart Contract Address:** None
- **Audit Date:** 2024-01-25
- **Audit Tools Used:** None
- **Auditors:** Barba

Executive Summary

This Security Review is the first one of the Cyfrin Security and Auditing course. A simple codebase is explored to give a introduction to audit process.

Methodology

In this research were applied technics of manual code review and tests suits.

Audit Findings

Critical Vulnerabilities

[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, severely breaking the functionality of the protocol.

Proof of Concept:

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

1. Create a locally running chain

```
anvil
```

2. Deploy the contract to the chain

```
forge script script/DeployPasswordStore.s.sol:DeployPasswordStore --rpc-url
http://localhost:8545 --private-key <PRIVATE_KEY> --broadcast
```

3. Run the storage tool We use `1` as storageSlot because that's the storage slot of `s_password` in the contract.

```
cast storage <contractAddress> <storageSlot> --rpc-url http://localhost:8545
```

You'll get an output that looks like this:

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

You can parse that hex to a string with:

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

And get and output of:

```
myPassword
```

Recommendation:

- ue to this, the overall architecture of the contract should be rethought. One could encrypt the password off-chain, and then store the encrypted password on-chain. This would require the user to remember another password off-chain to decrypt the password. However, you'd also likely want to remove the view function as you wouldn't want the user to accidentally send a transaction with the password that decrypts your password.

Status:

- None

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

Description:

- The `PasswordStore::setPassword` function 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.`

▶ Code

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

Impact:

- Anyone can set / change the password of the contract, severely breaking the contract intended functionality.

Proof of Concept:

- Add the following to the `PasswordStore.t.sol` test file.

▶ Code

```
    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);
    }
```

Recommendation:

Add an access control conditional to the `setPassword` function.

▶ Code

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

Status:

- None

Minor Observations

[I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect

Description:

▶ Code

The `PasswordStore::getPassword` function signature is `getPassword()` while the natspec say it should be `getPassword(string)`.

Recommendation:

- Remove the incorrect natspec line.
-

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

Status:

- None

Conclusion

The vulnerabilities found in this review were basic and obvious. However, were a great introduction to practices of code auditing.

Appendices