Protocol Audit Report

Version 1.0

lealCodes

January 12, 2024

PasswordStore Audit Report

lealCodes

Oct 19th, 2023

Prepared by: Leal

Table of Contents

Protocol Summary

CodeHawks First Flights #1

Disclaimer

lealCodes 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 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

Audit Details

Sponsor: First Flight #1

Dates: Oct 18th, 2023 - Oct 25th, 2023

Scope

./src/ – PasswordStore.sol

Roles

Owner - Only the owner may set and retrieve their password.

Executive Summary

After a time boxed security review 2 high issues were found.

Issues found

Number of findings:

- High: 2
- Medium: 0
- Low: 0

High

H-01. Anyone can see what the password is

Relevant GitHub Links

https://github.com/Cyfrin/2023-10-PasswordStore/blob/main/src/PasswordStore.sol#L14C4-L14C31

Summary

Just because a variable on a smart contract is labeled `private` it does not mean it can't be accessed. Someone can simply look at the storage slots of the deployed PasswordStore.sol contract and get the password.

Vulnerability Details

to illustrate this vulnerability first we can run `make anvil` to create a blockchain locally.

then on a new terminal we can deploy the contract using `make deploy` then run `cast storage "contract address"`

the last command will output the following:

```
| Name         | Type     | Slot | Offset | Bytes | Value
   | Contract                                |
|------------|---------|------|--------|-------|------------------------------------
| s_owner     | address | 0    | 0      | 20    |
   1390849295786071768276380950238675083608645509734
                              | src/PasswordStore.sol:PasswordStore |
```

```
| s_password | string  | 1    | 0      | 32    |
49516443757395204518384437876896412918898210405993719258753982441762571943956
 | src/PasswordStore.sol:PasswordStore |
```

As it can be seen by the table above a value is given for the `s_password` variable, converting that into hexadecimal we get: `6D7950617373776F7264` and then converting that into a string we get: `myPassword`

Impact

This issue has been listed as High, since anyone can see the value of `s_password`

Tools Used

Foundry & Manual Review

Recommendations

Store your password off-chain. Nothing on the blockchain is private.

H-02. Access Control - non-owner can set a new password

Relevant GitHub Links

https://github.com/Cyfrin/2023-10-PasswordStore/blob/main/src/PasswordStore.sol#L26C5-L29C6

Summary

in PasswordStore.sol the function `setPassword()` does not check that `msg.sender == owner` allowing anyone to be able to set a new password.

Vulnerability Details

Running the test shown below in PasswordStore.t.sol will illustrate that a non-owner can set whatever new password they wish.

```
// @audit test
function test_non_owner_can_set_password() public {
    vm.startPrank(address(29));
    string memory expectedPassword = "I_Love_CodeHawks";
    passwordStore.setPassword(expectedPassword);
    vm.stopPrank();

    // Using the owner account to see the password
    vm.startPrank(owner);
    string memory actualPassword = passwordStore.getPassword();
    assertEq(expectedPassword, actualPassword);
}
```

Impact

This has been classified as a high issue, as anyone can set a new password.

Tools Used

Foundry & Manual Review

Recommendations

Add a check in `setPassword()` to ensure `msg.sender == owner`, such as the one shown below.

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