

read.md



Protocol Audit Report

Greater Heights

December 18, 2023

Prepared by: Greater Heights Computer

Lead Security Research:

- Adebara Khadijat

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer

- Risk Classification
- 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.
 - * [H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the password
 - Informational
 - * [I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect

Protocol Summary

PasswordStore is a protocol dedicated to storage and retrieval of a user's password. The protocol is designed to be used by a single user, and is not designed to be used by multiple users. Only the owner should be able to set and access this password.

Disclaimer

The Greater Heights Computer 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
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

The findings described in this document correspond the following commit hash:

7d55682ddc4301a7b13ae9413095feffd9924566

Scope

```
./src/  
|__ PasswordStore.sol
```

Roles

- Owner: The user who can set the password and read the password.
- Outsiders: No one else should be able to set or read the password.

Executive Summary

__Review this protocol was not too easy, we found state variables *password deploy onto chain which enable anybody to view the data on-chain, missing access control vulnerability and wrong natspec which indicate pass a parameter onto getPassword function. __We spent 5 hours with Khadijat and Abimbola auditors using CLoc and Solidity Metrics tools.

Issues found

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

Findings

High

[H-1] Storing the password on-chain makes it visable 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


```
}
```

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);
}
```

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

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

Informational

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

Description:

```
/*
 * @notice This allows only the owner to retrieve the password.
 * @param newPassword The new password to set.
 */
function getPassword() external view returns (string memory) {}
```

The PasswordStore::getPassword function signature is getPassword() which the natspec says it should be getPassword(string).

Impact: The natspec is incorrect.

Recommended Mitigation: Remove the incorrect natspec line.

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