



Protocol Audit Report

Version 1.0

Therock Ani

May 2, 2024

Protocol Audit Report

Therock Ani

May 2, 2024

Prepared by: Therock Ani

Lead Security Researcher:

- Therock Ani

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
 - Scope
 - Roles
- Executive Summary
 - Issues found
- Findings
 - High
 - [H-1] Variables stored in storage on-chain are visible to anyone, and no longer private
 - * Likelihood & Impact
 - [S-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the password

- * Likelihood & Impact 2
- Informational
- [I-#] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect
 - * Likeihood & Impact 3

Protocol Summary

Password is a protocol dedicated to storage and retrieval of a user's passwords. 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 Therock_Ani 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

```
1 2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

```
1 ./src/  
2 #-- PasswordStore.sol
```

Roles

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

Executive Summary

Add some notes about how the audit went, types of things you found, etc

We spent X hours with Z auditors using Y tools. etc

Issues found

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

High

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 be accessed through the `PasswordStore : : getPassword` function, which is intended to be only called by the owner of the contract.

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

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

- ```
1 make anvil
```

- ```
1 make deploy
```

- ```
1 cast storage <ADDRESS OF THE CONTRACT> 1 --rpc-url http://127.0.0.1:8545
```

[illegible][illegible]

```
1 myPassword
```

Therock Ani

**Likelihood & Impact**

- Impact: HIGH
- Likelihood: HIGH
- Severity: HIGH

**[S-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.

```
1 function setPassword(string memory newPassword) external {
2 @> // @audit - There are no access controls
3 s_password = newPassword;
4 emit SetNetPassword();
5 }
```

**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

```
1 function test_anyone_can_set_password(address randomAddress) public {
2 vm.assume(randomAddress != owner);
3 vm.prank(randomAddress);
4 string memory expectedPassword = "myNewPassword";
5 passwordStore.setPassword(expectedPassword);
6
7 vm.prank(owner);
8 string memory actualPassword = passwordStore.getPassword();
9 assertEq(actualPassword, expectedPassword);
10 }
```

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

```
1 if(msg.sender != s_owner){
2 revert PasswordStore_NOTOwner();
3 }
```

## Likelihood & Impact 2

- Impact: HIGH
- Likelihood: HIGH
- Severity: HIGH

## Informational

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

**Description:** The `PasswordStore::getPassword` function's natspec incorrectly indicates a parameter `newPassword` that doesn't exist in the function signature. This discrepancy causes the natspec to be inaccurate and potentially misleading for developers.

```
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 incorrect natspec may mislead developers regarding the function's parameters.

**Recommended Mitigation:** Remove the incorrect natspec line.

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

## Likelihood & Impact 3

- Impact: HIGH
- Likelihood: NONE
- Severity: Informational/Gas/Non-crits

Informational: Hey, this isn't a bug, but you should know...