Protocol Audit Report

Version 1.0

Cyfrin.io

November 27, 2024

PasswordStore Audit Report

Brian

November 27, April

Prepared by: Cyfrin
Lead Auditors: Brian Har

- xxxxxxx

Table of Contents

Protocol Summary

Protocol does X, Y, Z

Disclaimer

Brian Har 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

Scope

Roles

Executive Summary

Issues found

Findings

High

[H-1] Storing 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 var 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: All people can read the private password, severely breaking the functionality of the protocol.

Proof of Concept: (proof of code) - very important!

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

Make anvil -> deploy -> run the storage tool (cast parse-bytes32-string).

Recommended Mitigation:

1. The overall architecture of the contract should be rethought.
2. 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.

[H-2] `PasswordStore::setPassword` function has no access controls, meaning a non-owner will be able to 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.`

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

Impact: Anyone can change the password

Proof of Concept:

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); // this will pass
}
```

Informational

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

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

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

Impact: Natspec is incorrect

Recommended Mitigation: Remove the incorrect natspec line

```

- * @param new Password The new password to be set
```