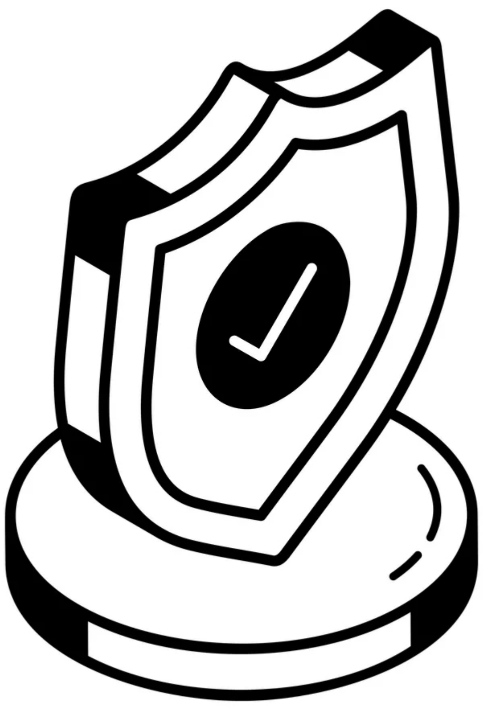

Protocol Audit Report

Version 1.0

YANPITRIK

July 10, 2024

Protocol Audit Report

Yan Pitrik

Juny 10, 2024

Prepared by: [Yan Pitrik] Lead Security Researcher:

- just me

Table of Contents

Protocol Summary

PasswordStore is a protocol which should be used to storing and retrieving owners password. Only the owner should be able to set new password and then access it.

Disclaimer

The author 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
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

Audit Details

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

Issues found

Severity	Number of issues
HIGH	2
MEDIUM	0
LOW	0
INFO/GAS	1
——-	————
TOTAL	3

Findings

High

[H-01] `PasswordStore::s_password;` Variables in storage are visible to anyone, even if visibility keyword is set to `private`. Password isnt private!

Description: On-chain data is visible to anyone and can be read directly from the blockchain. The `PasswordStore::s_password` variable is intended to be a private and only accessed through the `PasswordStore::getPassword` function. This function is intended to be only called by the owner of the contract.

Impact: Anyone can read private password which severly breaking the protocol functionality.

Proof of Concept: Below is the test case that shows how anyone can read the password from the storage.

Add this test function to the `PasswordStore.t.sol` file:

```
function test_anyoneCouldReadPassword() public {
        bytes32 passwordBytes = vm.load(address(passwordStore), bytes32
            (uint256(1)));
        bytes memory password = abi.encodePacked(passwordBytes);
        console.logBytes(password);
        console.log(string(password));
}
```

and then run test with command `forge test --mt test_anyoneCouldReadPassword -vvv`

Recommended Mitigation: You cannot store uncrypted password onchain! One way should be to encrypt password off-chain and store its encrypted version onchain.

[H-02] `PasswordStore::setPassword` function has no access control, anyone could reset the password

Description: The natspec of `PasswordStore::setPassword` function says that "only the owner is allowed to set a new password". However, the function is set to be external without any modifier of access controlling mechanism inside, which allows anyone to call it and set a new password.

```
    function setPassword(string memory newPassword) external {
@>      // @audit Missing access controls
        s_password = newPassword;
        emit SetNetPassword();
    }
```

Impact: Anyone can reset the password variable, severly breaking protocol functionality.

Proof of Concept: Here is proof of code which shows that anyone could reset the password.

Add this test function to the `PasswordStore.t.sol` file:

Show code

```
    function testFuzz_anyoneCouldResetPassword(address randomUser)
       public {
        //address randomUser = makeAddr("randomUser");
        vm.assume(randomUser != owner);

        vm.prank(owner);
        string memory oldPassword = passwordStore.getPassword();
        string memory newPassword = "myNewPassword";

        vm.prank(randomUser);
        passwordStore.setPassword(newPassword);

        vm.prank(owner);
        string memory actualPassword = passwordStore.getPassword();
        console.log(actualPassword);

        assertEq(actualPassword, newPassword);
        assert(keccak256(bytes(actualPassword)) != keccak256(bytes(
           oldPassword)));
    }
```

and then run test with command `forge test --mt test_anyoneCouldResetPassword -vvv`

Recommended Mitigation: Add access control to the top of the `setPassword` function

```
    function setPassword(string memory newPassword) external {

+      if (msg.sender != owner) {
+           revert PasswordStore__NotOwner();
+        }

        s_password = newPassword;
        emit SetNetPassword();
    }
```

Informational

[I-01] The `PassWordstore::getPassword` natspec indicates a param that doesnt exist, causing the natspec to be incorrect

Description:

The `PassWordstore::getPassword` function signature is `getPassword()` and the natspec say it should be `getPassword(string)`

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

Impact: The natspec is incorect.

Recommended Mitigation: Remove the incorrect natspec line.

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