Protocol Audit Report

Prepared by: 0xPexy

Table of Contents

Disclaimer

The 0xPexy 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

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

Commit hash in audited repository `7d55682ddc4301a7b13ae9413095feffd9924566`

Scope

```
src/
--- PasswordStore.sol
```

Protocol Summary

PasswordStore is a smart contract application for storing a password.

Key Features

- Users should be able to store a password and then retrieve it later.
- Others should not be able to access the password.

Roles

- Owner: is the only one who can set and get password from contract.

Executive Summary

Issues found

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

Findings

High

[H-1] Storing the 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 blockchain. The `PasswordSotre::s_password` varaible is intended to a private variable and only accessed through `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: Anyone can read the private password, severly breaking the functionality of the protocol.

Proof of Concept(Proof of Code): The below test case shows how anyone can read the password directly from the blockchain.

1. Create a local network

```
make anvil
```

2. Deploy contract

```
make deploy
```

3. Run the storage tool

```
cast storage <CONTRACT_ADDRESS> 1 --rpc-url localhost:8545
```

You'll get output:

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

You can parse it with:

```
cast parse-bytes32-string
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

Result:

```
myPassword
```

Recommended Mitigation: Due to this, the overall architecture of the contract should be rethought. One could encrypt the password on-chain. This would require the user to remember another password off-chain to decrypt the password.

[H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the password

Description: The `PasswordStore::setPassword` function is set to be `external`, however, `This function allow only the owner to set the password.`

```
function setPassword(string memory newPassword) external {
    // @audit - No access controls
    s_password = newPassword;
    emit SetNetPassword();
}
```

Impact: Anyone can change the private password, severly breaks intention of the contract.

Proof of Concept: Add the following to the `test/PasswordStore.t.sol`.

```
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 to the `setPassword` function.

```
if(msg.sender!=s_onwer) {
    revert
}
```

Informational

[I-1] The `PasswordStore::getPassword` natspec indicates a parameter doesn't exist

Description: The `PasswordStore::getPassword` signature is `getPassword()`, while the NatSpec says 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) {
    if (msg.sender != s_owner) {
        revert PasswordStore__NotOwner();
    }
    return s_password;
}
```

Impact: The natspec is incorrect.

Recommended Mitigation: Remove the incorrect NatSpec line.

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