Protocol Audit Report

Version 1.0

Cyfrin.io

January 21, 2024

PasswordStore Audit

0xShitgem

January 21, 2024

Prepared by: 0xShitgem Lead Auditors:

- Me

Table of Contents

Protocol Summary

PasswordStore is a protocol designed to store and retrieve securely password on-hain.

Disclaimer

The 0xShitgem 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:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

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

Roles

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

Executive Summary

Spent little time on audit

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 visible to anyone

Description: All data sotred on-chain is visible to anyone and can be read directly from the blockchain. The `PasswordStore::s_password` varaible is intended to be a private variable and only be accessed through the `PasswordStore::getPassowrd` function, which is intended to be only called by the owner of the contract.

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

Proof of Concept:

Below test case shows how anyone can read password directly

1. Create a locally running chain

```
1 make anvil
```

2. Deploy the contract to the chain

```
1 make deploy
```

3. Run storage tool

```
cast storage <contract_address> 1 --rpc-url http://127.0.0.1:8545
```

Output like this: `0x6d7950617373776f726400000000000000000000000000000000000000000014`

Parsing

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

And get an output of:

```
myPassword
```

Recommended Mitigation: Due to overall architecture of code, basically entire contract should be rewritten. The best way to implement function with access control is instead storing password use `s_owner` variable where you store address of EOA (or another contract). However, in this contract it's kinda impossible, beacuse here we only have two functions `PasswordStore::setPassword` and `PasswordStore::getPassword`

In this case we should encrypt password off-chain and encrypted password store on-chain. This would require user to remember password off-chain.

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

Description: The natspec of `PasswordStore::setPassword` function indicates only Owner should execute function. However there is no access control of this function meaning everyone can execute it.

```
function setPassword(string memory newPassword) external {
    s_password = newPassword;
    emit SetNetPassword();
}
```

Impact: Anyone can set/change password of the contract, severly breaking the contract intended functionality.

Proof of Concept: Add the follwoing to the `PasswordStore.t.sol` test file.

```
    function test_anyone_can_set_password(address randomAddress) public
        {
        vm.assume(randomAddress != owner);
        vm.prank(randomAddress);
```

```
4           string memory expectedPassword = "myNewPassword";
5           passwordStore.setPassword(expectedPassword);
6
7           vm.prank(owner);
8           string memory actualPassword = passwordStore.getPassword();
9
10          asserEq(actualPassword, expectedPassword);
11      }
```

Recommended Mitigation: Add an Access Control conditional to the `setPassword` function.

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

Or consider using `onlyOwner` modifier from `@openzeppelin` package

Medium

Low

Informational

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

Description:

```
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 say it should be `getPassword(string)`.

Impact: Natspec is incorrect

Recommended Mitigation: Remove the incorrect natspec line.

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

Gas