Protocol Audit Report

Version 1.0

github.com/dik654

December 31, 2023

Protocol Audit Report

dae ik kim

March 7, 2023

Prepared by: dik654 Lead Auditors: - dae ik kim

Table of Contents

Protocol Summary

Protocol does X, Y, Z

Disclaimer

The YOUR_NAME_HERE 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

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:

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

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

*Add some notes about how the audit

Issues found

Severtity	Number of issues found
High	2
Meidum	0
Low	0
Info	1
Total	3

Findings

High

[H-1] Storing the password on-chain makes it visable to anyone, and no longer private

Description: All data stored on-chain is visible to anyone, and can be read directly from blockchain.

Impact: Anyone can read the private password, severly breaking the functionlity 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 locally running chain

```
make anvil
```

2. Deploy the contract to the chain

```
make deploy
```

3. Run the storage tool We use 1 because that's the storage slot of `s_password` in the contract.

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

after this command, you can get this binary data 0x6d7950617373776f726400000000000000000000000000000000000000

and then you can get the password by parsing the data

```
cast parse-bytes32-string 0
   x6d7950617373776f726400000000000000000000000000000000000014
```

```
output: myPassword
```

Recommended Mitigation: Encrypt the password off-chain, and then store the encrypted password on-chain.

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

Description:

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

Impact: Anyone can set/change the password of the contract

Proof of Concept: 1. Add the following to the `PasswordStore.t.sol`

Code

```
    function test_anyone_can_set_password(address randomAddress) public
        {
        vm.prank(randomAddress);
        string memory expectedPassword = "myNewPassword";
        passwordStore.setPassword(expectedPassword);

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

```
8           assertEq(actualPassword, expectedPassword);
9       }
```

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

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

Informational

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

Description:

```
1       function getPassword() external view returns (string memory) {
2           if (msg.sender != s_owner) {
3               revert PasswordStore__NotOwner();
4           }
5           return s_password;
6       }
```

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

Impact: The natspec is incorrect

Recommended Mitigation: Remove the incorrect natspec line.

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

Gas