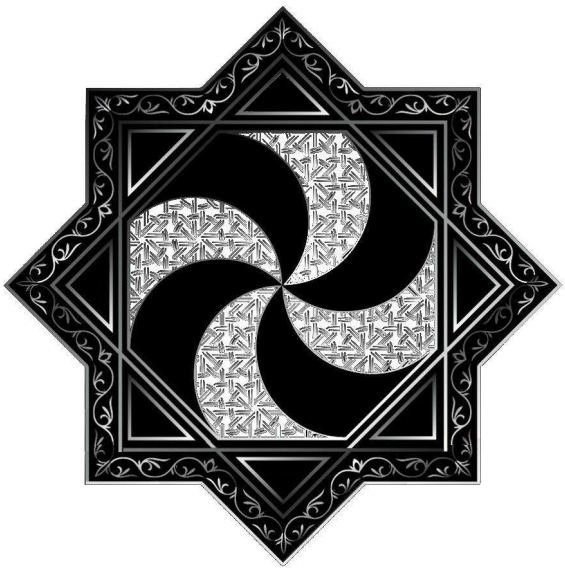

Audit Report

Version 1.0

Hebx

February 20, 2024

Audit Report

Hebx

February 20, 2024

Prepared by: Hebx Lead Auditors: - Ihab Heb

Table of Contents

Protocol Summary

PasswordStore is a protocol dedicated to storage and retrieval of a user's passwords. The protocol is designed to be used by a single user, and is not designed to be used by multiple users. Only the owner should be able to set and access this password.

Disclaimer

The Hebx 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
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

The findings described in this document correspond the following commit hash:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

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

Roles

- Owner: Is the only one who should be able to set and access the password.

For this contract, only the owner should be able to interact with the contract.

Executive Summary

Issues found

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

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 the blockchain, The `PasswordStore:s_password` variable is intended to be a private variable and only accessed through the `PasswordStore:getPassword` function, which is intended to be only called by the owner of the contract.

We show one 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: run a local anvil chain `make anvil` Deploy the contract `make deploy`

`cast storage 0x5FbDB2315678afecb367f032d93F642f64180aa3 1 --rpc-url http:127.0.0.1:8545` You get the password in the storage slot at index 1 in bytes32 format

`cast parse-bytes32-string 0x6d7950617373776f726400000000000000000000000000000000000000000000` You get `myPassword`

Recommended Mitigation: Due to this, the overall architecture of the contract must be rethink, one could encrypt the password off-chain and then store the encrypted password on-chain, this would require the user to remember the password to decrypt their main password. However, you should also remove the view function as you wouldnt' want the user to accidently send a transaction with the password the decrypt the main password

[H-2] `PasswordStore::setPassword` is callable by anyone

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.`

```
1     function setPassword(string memory newPassword) external {
2 @>      // @audit - There are no access controls here
3         s_password = newPassword;
4         emit SetNetPassword();
5     }
```

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

Proof of Concept:

Add the following to the `PasswordStore.t.sol` test suite.

```
1 function test_anyone_can_set_password(address randomAddress) public {
2     vm.prank(randomAddress);
3     string memory expectedPassword = "myNewPassword";
4     passwordStore.setPassword(expectedPassword);
5     vm.prank(owner);
6     string memory actualPassword = passwordStore.getPassword();
7     assertEq(actualPassword, expectedPassword);
8 }
```

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

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

```
3 }
```

Informational

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

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 natspec for the function `PasswordStore::getPassword` indicates it should have a parameter with the signature `getPassword(string)`. However, the actual function signature is `getPassword()`.

Impact: The natspec is incorrect.

Recommended Mitigation: Remove the incorrect natspec line.

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