



Protocol Audit Report

Version 1.0

Vesko.io

May 16, 2024

Protocol Audit Report

Vesko.io

May 16, 2024

Prepared by: Vesko Lead Auditors: - Veselin VACHkov

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
 - Scope
 - Roles
 - Issues found
- Findings
 - High
 - * [H-1] Storing the password on-chain makes it visible to anyone, and no longer private
 - * [H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the password
 - Informational
 - * [I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect

Protocol Summary

Protocol does X, Y, Z

Disclaimer

The Veselin's 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.

Scope

```
1 ./src/  
2 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.

Issues found

Sevterity	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, 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 called only 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, severely 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 locally running chain

```
1 make anvil
```

2. Deploy the contract to the chain

```
1 make deploy
```

3. Run the storage tool

```
1 cast storage 0x5FbDB2315678afecb367f032d93F642f64180aa3
```

You get a similar output: `0x6d7950617373776f726400`

Recommended Mitigation: Add a conditional to the `setPassword` function.

```
1  if(msg.sender != s_owner){
2      revert PasswordStorage_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
6      // @audit there is no param to be passed (documentation fix)
7      function getPassword() external view returns (string memory) {
```

The `PasswordStore::getPassword` function signature is `getPassword()` which the natspec says 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.
```