Protocol Audit Report

Version 1.0

January 31, 2025

PasswordStore Protocol Audit

scofield Idehen

January 31, 2025

Prepared by: Black Spider Lead Auditors:

- xxxxxxx

Table of Contents

Protocol Summary

Passwordstore is a protocol dedicated to storage and retrivial 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 shoud be set and able to access the password.

Disclaimer

The Black_spider 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 to the following commit hash

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4
```

Scope

```
./src/
PassowrdStore.sol
```

Roles

- Owner: the user who set the password and read the password
- Outsider: No one else should be able to set or read the password

Executive Summary

The Passwordsol file was a 45 line code base with two fucntion and a constructor.

While reviwing the codebase, we found 3 issues which show the codebase was succibale to attack and should be fixed immediality.

Issues found

Severity	NUmber of issues Found
High	2
Medium	0
Low	0
Info	0
Total	2

Findings

High

[s-1] (storing the password on-chain is visable to anyone, and no longer private)

Description: All data on chain is visable to anyone and can be read directly from the blockchain `Passwordstore::s_password` virable is intended to be private and should be called from the `Passwordstore::getPassword`

Impact: since anyone can access the password then it breaks the functionality of the 'Passwordstore and the contract.

Proof of Concept:

the below test case shows anyone can access the password

1. Create a local running chain

```
make anvil
```

2. Deploy the contract to the chain

```
make deploy
```

3. Run the storage tool

we use 1 beacuse thats the storage slot of s_password in the contract.

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

0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0 You will get an output like this

0x6d7950617373776f726400000000000000000000000000000000000000000014

0x6d7950617373776f726400000000000000000000000000000000000000000014

you can then parse that hex to a string

```
cast parse-bytes32-string 0x
```

And get an output of

```
mypassword
```

Recommended Mitigation: Dues to this, the overall structure should be redefined.

[s-1] (`Password::setPassword` has no access control, meaning a non-owner can change the password)

Description: The `PasswordStore::setPassword` function is set to be an `external` function, however the netspec of the function and overall purpose of the smart contract is that `this function allows only the owner to set a new password`

```

 function setPassword(string memory newPassword) external {
        // if (msg.sender != s_owner) {
```

```
4           //     revert PasswordStore__NotOwner();
5           // }
6           s_password = newPassword;
7           emit SetNewPassword();
8       }
```

Impact: Anyone can set/change the password therby breaking the password store.

Proof of Concept:

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

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