Finishers Guild Audit Report

Version 1.0

successaje.github.io

February 18, 2025

Finishers Guild Audit Report

successaje.github.io

February, 2025

Prepared by: Finishers Guild Lead Auditors: - Success Aje

Table of Contents

Protocol Summary

PasswordStore Protocol allows you to store a private password that others won't be able to see. It allows owner to be able to get and retrieve their saved password.

Disclaimer

The Finishers Guild 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 folllowing commit hash: **

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

Scope

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

Executive Summary

This report provides a detailed security assessment of the ProjectStore smart contract, identifying critical vulnerabilities that could compromise its integrity, security, and functionality. The assessment evaluates key risk areas, including access control, data exposure, and contract logic flaws.

Our findings highlight multiple security risks, such as storage visibility issues, unauthorized access to sensitive functions, and documentation inconsistencies that could lead to exploits. Each vulnerability is accompanied by a proof of concept and recommended mitigation strategies to enhance the contract's security.

By implementing the suggested security improvements, the project can mitigate risks, protect user assets, and ensure robust smart contract functionality.

Issues found

Severity	Number of issues found
High	2
Medium	0
Low	0
Info	1
Total	3

Findings

High

[H-1] Variables Stored in Storage Are Visible to Anyone, Making the Password Not Actually Private

Description:
All data stored on-chain is publicly accessible and can be read directly from the blockchain. The `PasswordStore::s_password` variable is intended to be private and accessible only through the `PasswordStore::getPassword()` function, which should only be called by the contract owner.

However, it is possible to retrieve this data off-chain, exposing the password.

Impact:
Anyone can read the supposedly private password, severely compromising the protocol's security.

Proof of Concept:

The following test case demonstrates how anyone can retrieve the password directly from the blockchain:

- `make anvil`
- `make deploy`
- `cast storage <address> 1 --rpc-url 127.0.0.1:8545`
- `cast parse-bytes32-string 0x6d7950617373776f726400000000000000000000000000000000`

The retrieved output will reveal the password, which was intended to be private.

Recommended Mitigation: Store the data off-chain using a secure storage solution. Encrypt the password before storing it on-chain to prevent direct retrieval.

[H-2] Anybody can set the password

Description: There are no access controls on the `setPassword()` function, allowing any user to change the password.

Impact: Unauthorized users can modify the stored password, potentially disrupting the protocol's intended functionality.

Proof of Concept: The following test case in `PasswordStore.t.sol` demonstrates how an arbitrary address can modify the password:

Code

```
    function test_anybody_can_set_password(address _address) public {
        vm.assume(_address != owner);
        vm.prank(_address);
        string memory newPassword = "myNewPassword";
        passwordStore.setPassword(newPassword);

        vm.prank(owner);
        string memory actualPassword = passwordStore.getPassword();
        assertEq(actualPassword, newPassword);
    }
```

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

```
    if (msg.sender != s_owner){
        revert PasswordStore_NotOwner()
    }
```

Informational

[I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesnt exist , causing the natspec to be incorrect.