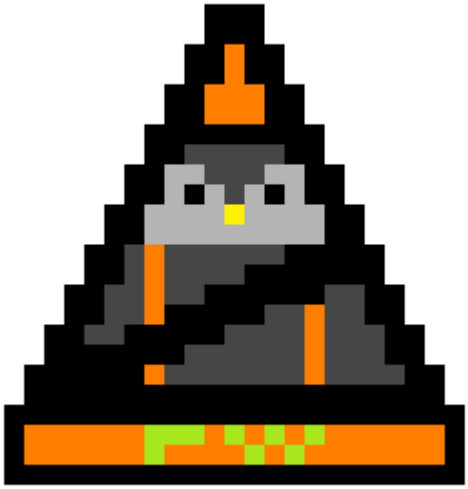

Protocol Audit Report

Prepared by: Rong Wei Zhang

Table of Contents

Protocol Summary

Protocol does X, Y, Z

Disclaimer

The auditor 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: Commit Hash:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

```
./src/
└── PasswordStore.sol
```

Roles

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

Executive Summary

Issues found

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

Findings

High

[H-1] Storing password directly on-chain makes it visible to everyone and is not private

Description: All data stored on-chain can be viewed by anyone and is directly readable from the blockchain. The `PasswordStore::s_password` variable is not intended to be read or viewed by anyone except the contract owner using `PasswordStore::getPassword` function.

Impact: The owner's password may be read by anyone and used maliciously against the owner.

Proof of Concept:

The following test case shows how anyone can read the password directly from the blockchain.

1. Run a local Anvil chain using `anvil` command
2. Deploy PasswordStore.sol to Anvil using

```
make deploy
```

3. Read the memory of `PasswordStore::s_password` using

```
cast storage <PasswordStore contract address>
```

4. Convert binary data into string using

```
cast parse-bytes32-string <binary representation obtained in step 3>
```

Recommended Mitigation: Encrypt the password off-chain before storing it on-chain. However, this would require the user to remember another password off-chain and would require restructuring the project.

Likelihood & Impact:

- Impact: HIGH
- Likelihood: HIGH
- Severity: HIGH

[H-2] `PasswordStore::setPassword` has no access control, which allows everyone to set the password

Description: The `PasswordStore::setPassword` function is declared as `external` (publicly accessible), but the natspec comment says `This function allows only the owner to set a new password.` However, there is no access control implemented in the function.

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

Impact: Anyone can set or modify the password, making the password storage meaningless and allowing attackers to lock out the owner

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

▶ Code

```

```

```
function test_anyone_can_set_password(address attacker) public {
        vm.prank(attacker);
        string memory expectedPassword = "myNewPassword";
        passwordStore.setPassword(expectedPassword);
        vm.prank(owner);
        string memory actualPassword = passwordStore.getPassword();
        assertEq(actualPassword, expectedPassword);
    }
```

Recommended Mitigation: Add access control such as:

```
if(msg.sender != i_owner) revert();
```

Likelihood & Impact:

- Impact: HIGH
- Likelihood: HIGH
- Severity: HIGH

Informational

[I-1] Natspec documentation for `PasswordStore::getPassword` indicates a `newPassword` parameter that doesn't exist

Description:

```
/*
     * @notice This allows only the owner to retrieve the password.
@>   * @param newPassword The new password to set.
     */
    function getPassword() external view returns (string memory) {
```

The `PasswordStore::getPassword` function does not take a `newPassword` parameter, so this line in the documentation is incorrect.

Impact: The Natspec documentation is incorrect and misleading.

Recommended Mitigation: Remove the incorrect parameter documentation line.

```
/*
     * @notice This allows only the owner to retrieve the password.
-     * @param newPassword The new password to set.
     */
```

Likelihood & Impact:

- Impact: None
- Likelihood: HIGH
- Severity: Informational/gas/Non-crits