\begin{titlepage} \centering \begin{figure}[h] \centering \includegraphics[width=0.5\textwidth]{logo.pdf} \end{figure} \vspace* {2cm} {\Huge\bfseries Protocol Audit Report\par} \vspace{1cm} {\Large Version 1.0\par} \vspace{2cm} {\Large\itshape Cyfrin.io\par} \vfill {\large \today\par} \end{titlepage}

\maketitle

Prepared by: Dionis Xhaferi Lead Auditors:

- xxxxxxx

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
  - Scope
  - Roles
- Executive Summary
  - Issues found
- Findings
- High
- Medium
- Low
- Informational
- Gas

Protocol Summary

Protocol does X, Y, Z

Disclaimer

The DX 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

Scope

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.

Executive Summary

Issues found

High - 2 Medium - 0 Low - 0 Info - 1 Total - 3

Findings

High

[H-1] Storing password onchain makes it visible to anyone and no longer private (Root Cause + Impact)

Description:

All data stored onchain is visible to anyone and can be directly read 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 contracts

We show one such 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:

1. createa locally running chain

   `make anvil`

2. Deploy the contract to the chain

`make deploy`

3. Run the storage tool

   We use 1 because that is the storage slot of s_password in the contract

   `cast storage ,ADDRESS_HERE> 1 --rpc-url http://127.0.0.1:8545`

   You get an ouput that looks like this

   `0x6d7950617373776f726400000000000000000000000000000000000000000014`

   You can then parse that hex to a string with:

   `cast parse-bytes32-string 0x6d7950617373776f726400000000000000000000000000000000000000000014`

   and you get the output :

   `myPassword`

Recommended Mitigation:

```
Due to this, the overall architecture of the contract should be rethought. one could encrypt the password
off-chain, and then store the encrypted password onchain. This would require the user to remember another
password offchain to decrpyt the password. however, you'd also likely want to remove the view function as
you wouldn't want the user to accidentally send a transaction with the password that decrypts your password.
```

[H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the password.

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.`

Impact:

Anyone can set/change the password of the contract, breaking the code functionality.

Proof of Concept:

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

```
    function test_anyone_can_set_password(address randomAddress) public {
        vm.assume(randomAddress != owner);
        vm.prank(randomAddress);
        string memory expectedPassword = "myNewPassword";
        passwordStore.setPassword(expectedPassword);

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

Recommended Mitigation:

Add an access control conditional to the `setPassword` function.

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

Medium

Low

Informational

[I-1] `PasswordStore::getPassword` natspec indicates a parameter that doesnt exist

Description:

The `PasswordStore::getPassword` function signature is `getPassword();` while the natspec says it should be `getPassword(string)`

Impact:

the natspec is incorrect.

Recommended Mitigation:

Remove the incorrect natspec line.

```
-   * newPassword The new password to set
```

Gas