Protocol Audit Report

Version 1.0

Fels21

June 30, 2024

Protocol Audit Report

Fels21

June 23, 2024

Prepared by: Fels21 Lead Security Researcher: - Fels21

Table of Contents

Protocol Summary

PasswoedStore is a smart contract applicatoin for storing a password. Users should be able to store a password and then retrieve it later. Others should not be able to access the password.

Disclaimer

The 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

- Commit Hash: 7d55682ddc4301a7b13ae9413095feffd9924566

Scope

```
./src/
  PasswordStore.sol
```

Roles

- Owner: The user who can set the password 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	1
Info	1
Gas Optimizations	0
Total	0

Findings

High

[H-1] Variables stored on-chain are visible to everyone, and they are not actually “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 smart contract.

We show one 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: The below test case shows how anyone can read the password directly from the blockchain.

1. Create a locally running chain: `bash make anvil`
2. Deploy the contract on the chain: `go make deploy`
3. Run the storage tool: `arduino cast storage {sc_address} {num_slot} --rpc-url http://127.0.0.1:8545`

You will get an output like this:

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

Now you can parse that hex to a string with:

```
cast parse-bytes32-string 0
    x6d7950617373776f726400000000000000000000000000000000000000000014
```

And the result output will be:

```
myPassword
```

Recommended Mitigation: All protocol does not have sence due implemtation of the pasword

[H-2] `PasswordStore::setPassword` has no acces controls, meamning that non-ownwer conuld change the password

Description: The `PasswordStore::setPassword` it set to be `external` function, however, the natspec indicates a `This function allows only the owner to set a new password.`

```
    function setPassword(string memory newPassword) external {
        // @audit there are no acces controls
        s_password = newPassword;
        emit SetNetPassword();
    }
```

Impact: Anyone can set or change the contract, severely breaking the contract's intended 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);

    // Setting password as random user
    vm.prank(randomAddress);
    string memory expectedPassword = "randomPassword";
    passwordStore.setPassword(expectedPassword);

    // Getting password as original owner
    vm.prank(owner);
    string memory actualPassword = passwordStore.getPassword();
    assertEq(actualPassword, expectedPassword);
}
```

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

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

Informational

[I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect

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) {
```

Impact: The natspec is incorrect.

Recommended Mitigation: Remove the incorrect natspec line:

```
-     * @param newPassword The new password to set.
```