Protocol Audit Report

Prepared by: YASIR IQBAL

Prepared by: YASIR IQBAL Lead Auditors:

Table of Contents

Protocol Summary

Security Audit Review by using tools like foundry, forge, anvil, cast, anvil, Solidity_Matric, slither, aderyn in 4 hours for manual code reading and finding Risks

Disclaimer

The YASIR_IQBAL 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

	Impact		
Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

Commit Hash: 2e8f81e263b3a9d18fab4fb5c46805ffc10a9990

Scope

./src/ └── PasswordStore.sol

Roles

Owner - Only the owner may set and retrieve their password

Executive Summary

Issues found

Severity	No of Issues Found
High	2
Mediam	0
Low	0
Informational	1
Total	3

Findings

High

[High-01] Storing Password on-chain as a private varilabe, is public to any one.

Description: All data stored on chain is visibe to anyone. So storing `PasswordStorne::s_password` variabe intended to be proviate, can ony be get by `PasswordStorne::getPassword` function which is intended to be call by onlyOwner we have to show any way to read data on chain which is defined as private

Impact: Any one can read the private password and severy braking the functionality of the protocole.

Proof of Concept: (proof of the code)

this is the test case inwhich any one can read password stored on-chain

1. Run local chain to deploy

```
anvil
```

2. Deploy contract by running scripts

```
forge script ./script/DeployPasswordStore.s.sol --rpc-url http://127.0.0.1:8545
```

contract PasswordStore 0x34A1D3fff3958843C43aD80F30b94c510645C316

3. read storage variable on deployed contract on specific Storage slot of rpc-url of chain

```
cast storage 0x34A1D3fff3958843C43aD80F30b94c510645C316 1 --rpc-url
http://127.0.0.1:8545
```

You will get output

```
0x111cc700d4d89afabec98a8af70df7dfb7ab7be3344ca39b2289344928b18840
```

4. parse the output hash from bytes32 to string to read password

```
cast parse-bytes32-string "output of above command"
```

```
monuPassword
```

Recommended Mitigation: Due to this , complete architecher of the contract rethought. User can encypit password off-chain and store encypted password on-chain. for that use need to remember another password to decrypit. Aso need to remove view functions, as use can accidenty send password with transaction to decrypt the password.

[High-02] `PasswordStore::setPassword` no access control, means non-Owner can change to password.

Description: The `PasswordStore::setPassword` function is an `Exteranl` function, however , the naspec of of the function and whole purpose of the smart contract is that ` The function allow only the owner to change password.

```
    function setPassword(string memory newPassword) external {
@>      // @Audit There are no Access control
```

```
        s_password = newPassword;
        emit SetNewPassword();
    }
```

Impact: Any one can change/set password of the contract, severily breaking the functionality.

Proof of Concept: Adding following code to th `PasswordStore.t.sol` test file

▶ Test Code

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

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

▶ TestOutput

```
Ran 1 test for test/PasswordStore.t.sol:PasswordStoreTest
[PASS] test_anyone_can_change_password(address) (runs: 256, μ: 23517, ~: 23517)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 39.35ms (36.17ms CPU
time)
```

Recommended Mitigation: Add an access control function in `PasswordStor::setPassword` as below

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

Informational

[Info-01] The `PasswordStore::getPassword` function not have newPassword variable, while Natspec is wrong.

Description: The Natspec is wrongly define the newPassword

```
    /*
     * @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:

Proof of Concept:

The `PasswordStore::getPassword` function signature is getPassword(), while natspec show it should be getPassword(string newPassword)

Recommended Mitigation: Need to remove the natspec as below

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