Audit Report

Version 1.0

Zkillua.io

Password Store Audit Report

Zkillua.io

Dec 20, 2023

Prepared by: Zkillua Lead Auditors: - Zkillua

Table of Contents

See table

Protocol Summary

PasswordStore is a protocol dedicated to storage and retrieval of a user's passwords. The protocol is designed to be used by a single user, and is not designed to be used by multiple users. Only the owner should be able to set and access this password.

Disclaimer

The Zkillua 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
Likelihood	High	H	H/M	M
	Medium	H/M	M	M/L
	Low	M	M/L	L

Scope

commit Hash: 2e8f81e263b3a9d18fab4fb5c46805ffc10a9990 src/ — PasswordStore.sol

Roles

Owner: Is the only one who should be able to set and access the password

Issues found

Severity	Number of issues found
High	2
Medium	0
Low	0
Info	0
Gas Optimizations	0
Total	0

Findings

[H-1]AnyOne Can change password by calling `PasswordStore::setPassword`

Description: The passwordStore contract assumes only owner can set the password, but due to absence of access modifiers, anyone can alter the set password.

Impact: Since anyone, inluding malicious actors, can set the password, this opens up to the possibility that, depending on the context, these unsantisied and potentially malicious strings could be dangerous. This negates the use of Contract.

Proof of Concept:

```
1   function test_anyone_can_set_password() public{
2           address  attacker=makeAddr("attacker");
3           vm.startPrank(attacker);
4           string memory newPassword="attackerPassword";
5           passwordStore.setPassword(newPassword);
6           console.log("The attacker successfully set the password:" ,
                newPassword);
7       }
```

Recommended Mitigation:

Using If statement:

```
1       function setPassword(string memory newPassword) external {
2           s_password = newPassword;
3   ++        if(msg.sender!=owner){
4   ++            error PasswordStore__NotOwner();
5   ++            }
6           emit SetNetPassword();
7       }
```

Usage of access Modifiers like from openzeppelin:

```
1    // @audit import the ownable contract from OpenZeppelin
2   + import "@openzeppelin/contracts/ownership/Ownable.sol";
3
4    // @audit inherit from the Ownable contract
5   + contract PasswordStore is Ownable{
6       error PasswordStore__NotOwner();
7
8       address private s_owner;
9       string private s_password;
10
11      event SetNetPassword();
12
13  ++   constructor() Ownable() {
```

```
++        s_owner = msg.sender;
++    }
}
```

[H-2] Anyone can read the set Password

Description

It is a misconception that since password is declared as private, nobody would be able to read the password other than owner. Everything on blockchain is public, and the set variables can be read by the attacker

Impact

This vulnerability completely compromises the protocol, exposes the sensitive data.

Proof of Concept

```

function test_anyone_can_see_the_password() public{
        string memory victimpassword="Victim password";
        vm.startPrank(owner);
        passwordStore.setPassword(victimpassword);

        //attacker exploit code
        uint256 storage_slot_value=1; //since owner slot is 0
        bytes32 slot_data=vm.load(
            address(passwordStore),
            bytes32(storage_slot_value)
        );

        //convert bytes into string

        string memory visible_password=string(abi.encodePacked(
           slot_data));

        console.log("anyone can read this password ",visible_password);



    }
```

Recommended Mitigation

Since everything on blockchain is public, to store sensitive information use additional encryption techniques. Never store sensitive data on public blockchain without encrypting.