Protocol Audit Report

Version 1.0

cosminmarian53

April 11, 2025

Protocol Audit Report

cosminmarian53

April 10, 2025

Prepared by: cosminmarian53
Lead Auditors: - cosminmarian53

Table of Contents

Protocol Summary

The PasswordStore contract assumes that only the owner can set the password. The setPassword() function modifies the s_password storage variable, where the password is set, but doesn't include access control meaning that anyone, including a malicious actor, can reset the owner's password.

Number of findings:

- High: 2
- Medium: 0
- Low: 1

Disclaimer

The lead auditor cosminmarian53 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

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

Audit Details

Roles

- Owner: The owner of the PasswordStore contract who can store the password.
- Outsiders: The other users which can interact with the protocol.

Executive Summary

This report identifies several key vulnerabilities within the smart contract implementation. The major findings include:

- **[H-1]**: Private variables, such as passwords, stored on-chain can be read by anyone despite Solidity's visibility modifiers.
- **[H-2]**: Lack of access controls in the `setPassword` function allowing any user to modify the stored password.
- **[I-1]**: An incorrect natspec parameter in the `getPassword` function documentation.

These issues compromise the expected confidentiality and functionality of the protocol. The detailed findings and recommendations follow in the sections below.

Issues found

- Exposure of private variables on-chain (**HIGH** severity).
- Missing access control for password updates (**HIGH** severity).
- Documentation (natspec) inconsistency in the password retrieval function (**Informational**).

Findings

[H-1] Private Variables Like Passwords Are Visible On-Chain Despite Visibility Modifiers

Likelihood and Impact:

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

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 private and only accesible through the `PasswordStore::getPassword` function, which is intended to be only called by the owner of the contract.

We show one such method of reading any data off chain bellow.

Impact: Anyone can read the private password, severly breaking the functionality of the protocol.

Proof of Concept:(Proof of Code)

The test case bellow shows that anyone can read the password from the blockchain. Create a locally running chain

```
make anvil
```

Deploy the contract to the chain

```
make deploy
```

Run the storage tool

We use 1 because that's the storage slot of s_password in the contract.

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

You'll get an output that looks like this:

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

You can then parse that hex to a string with:

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

And get an output of:

```
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 on-chain. This would require the user to remember another password off-chain to decrypt the stored password. However, you're also likely want to remove the view function as you wouldn't want the user to accidentally send a transaction with this decryption key.

[H-2]`PasswordStore::setPassword` has no access controls, which means that a non-owner could change the password to whatever he desires

Likelihood and Impact:

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

Description: The `PasswordStore::setPassword` function is set to `external`, however, the natspec of this function and overall purpose of the contract is that **ONLY THE OWNER CAN CHANGE THE PASSWORD**

```
//  @audit-high: Any user can call this function and set a password.
    //  missing access control
    function setPassword(string memory newPassword) external {
        s_password = newPassword;
        emit SetNetPassword();
    }
```

Impact: Anyone can change the password of the contract, severly breaking the intended functionality.

Proof of Concept: The following test proves that anyone can set/change the password.(should be added to `PasswordStore.t.sol`):

```
    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 `PasswordStore::setPassword` function

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

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

Likelihood and Impact:

- Impact: NONE
- Likelihood: HIGH
- Severity: Informational/Gas/Non-critical

Description:

```
    /*
     * @notice This allows only the owner to retrieve the password.
     // @audit there is no newPassword parameter in the function
    @>--- * @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.
```