Protocol Audit Report

Version 1.0

Shrid Mishra

March 5, 2025

Protocol Audit Report

Shrid Mishra

March 5, 2025

Prepared by: Shrid Mishra
Lead Auditors:
- Shrid Mishra

Table of Contents

- **Description:**
- **Impact:**
- **Recommended Mitigation:**
- **Suggested Correction:**

Protocol Summary

PasswordStore is a protocol dedicated to storage and retrieval of a user's password.

Disclaimer

The Shrid Mishra 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:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

```
./src/ pandoc report.md -o report.pdf --from markdown --t
emplate=eisvogel --listings
#-- PasswordStore.sol
```

Roles

-Owners: Set and read password. -Outsiders: Cannot set and read password.

Executive Summary

We spend 10 hours with 1 auditor.

Issues found

Severity	Number of issues found
High	2
Medium	0
Low	0
Info	1
Total	3

Findings

High

[H-1] storing the password on-chain makes it visible to anyone and no longer 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 can be accessed through a `PasswordStore::getPassword` function, which is intended to be called by the owner of the contract.

We show one such method of reading data off chain below.

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

Proof of Concept:

1. Create a local chain

```
make anvil
```

2. Deploy the contract

```
make deploy
```

3.

```
cast storage 0x5FbDB2315678afecb367f032d93F642f64180aa3 1 --rpc-url
    http://127.0.0.1:8545
```

4.

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

Recommended Mitigation: Due this you, overall contract should be rethought, encrypt the password off-chain.

[H-2] `PasswordStore::setPassword` has no access control meaning a non-owner could change it.

Description: The `PasswordStore::setPassword` is set to be `external`, however the purpose of the smart contact is that `This function only allows owner to set new password`.

```

  function setPassword(string memory newPassword) external {
   @>  // @audit - There are no access controls
        s_password = newPassword;
        emit SetNetPassword();
    }
```

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

Proof of Concept: Add this to `PasswordStore.t.sol` test file.

Code

```
    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 access control to `setPassword` function.

Informational

[I-1] Incorrect NatSpec Parameter in `PasswordStore::getPassword`

Description: The NatSpec documentation for the `PasswordStore::getPassword` function incorrectly references a parameter `newPassword` that does not exist in the function signature. The actual function signature is:

```
function getPassword() external view returns (string memory)
```

However, the NatSpec comment suggests the function takes a `newPassword` parameter, which is misleading and incorrect.

Impact:

- The documentation misrepresents the function's behavior.
- Developers might assume the function requires a parameter, leading to confusion.
- Incorrect documentation can cause issues during contract integration and maintenance.

Recommended Mitigation: Remove the incorrect `@param` line from the NatSpec documentation.

Suggested Correction:

```
 /*
  * @notice This function allows only the owner to retrieve the
     password.
- * @param newPassword The new password to set.
```

```
     */
    function getPassword() external view returns (string memory);
```