Password-Store Audit Report

Version 1.0

Ramprasad goud

March 16, 2025

Password-Store Audit Report

Ramprasad goud

March 16, 2025

Prepared by: Ramprasad goud

Executive Summary

This audit was conducted to evaluate the security of the PasswordStore contract. The scope included a manual review of the code, static analysis, and fuzz testing to identify vulnerabilities. Key findings revealed critical security risks, including the exposure of sensitive data and inadequate access controls.

Critical Security Risks Identified: - On-chain password storage in plaintext, leading to potential data leakage. - Lack of access control in the `setPassword` function, allowing unauthorized users to modify passwords.

Table of Contents

Scope & Objectives

The audit focused on the following objectives: - **Testing Methods:** Manual code review, static analysis with tools like Slither, and fuzz testing. - **Contracts Audited:** - `PasswordStore.sol`: Responsible for storing and retrieving user passwords.

Disclaimer

This audit was conducted by Ramprasad goud in an individual capacity. All efforts were made to identify vulnerabilities in the code within the given time constraints; however, no guarantees are provided regarding the completeness or security of the system. This audit is not an endorsement of the underlying business or product. The review focused solely on the security aspects of the Solidity implementation of the contracts and was conducted within a limited time frame.

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

The findings described in this document corresponded the following commit hash:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Findings

[H-1] On-Chain Password Storage is Publicly Accessible (High)

Summary: The contract stores passwords on-chain in plaintext, making them accessible to anyone via blockchain explorers.
Impact: Private data leakage, loss of confidentiality.
Proof of Concept (PoC):

```
1. Create a locally running chain.
   make anvil

2. Deploy the contract on the chain.
   make deploy

3. Run the storage tool.
   cast storage <Address_here> 1 --rpc-url http://127.0.0.1:8545
```

You will get an output that looks like this:

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

You can parse that hex to a string with:

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

And get an output of:

```
myPassword
```

Recommendation: Encrypt the password off-chain before storing on-chain. This would require the user to remember another password off-chain to decrypt the password. Additionally, consider removing the view function to prevent accidental exposure.
Tooling Evidence: Slither, Foundry.

[H-2] Lack of Access Control in `PasswordStore::setPassword` (High)

Summary: The `PasswordStore::setPassword` function is defined as `external`, allowing anyone to invoke it, contrary to the intended access restrictions.
Impact: Any user can set or change the contract's password, severely breaking its intended functionality.
Proof of Concept (PoC): Add the following to the `PasswordStore.t.sol` to test:

```
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);
}
```

Recommendation: Implement an access control condition in the `setPassword` function:

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

Tooling Evidence: Foundry.

[I-3] Incorrect Natspec in `PasswordStore::getPassword` (Informational)

Summary: The natspec documentation for `PasswordStore::getPassword` incorrectly indicates a non-existent parameter.
Impact: The natspec is misleading and could confuse developers using the contract.
Proof of Concept (PoC): The incorrect natspec line is:

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

Recommendation: Remove the incorrect natspec line to ensure clarity and accuracy in the documentation.
Tooling Evidence: Manual review.

Fix Review

If fixes are proposed later, a section will be added to verify the remediation of identified issues.

Conclusion

The findings indicate significant vulnerabilities that need to be addressed to ensure the security and proper functionality of the PasswordStore contract.