Protocol Audit Report

Version 1.0

CryptoBog

March 4, 2024

PasswordStore Protocol Audit Report

CryptoBog

March 4, 2024

Prepared by: [CryptoBog] Lead Auditors: - xxxxxxx

Table of Contents

Protocol Summary

Protocol does password storing and accessing functionality for users of the protocol.

Disclaimer

The YOUR_NAME_HERE 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: 7d55682ddc4301a7b13ae9413095feffd9924566

In Scope:

```
./src/
-- PasswordStore.sol
```

Solc Version: 0.8.18 Chain(s) to deploy contract to: Ethereum

Scope

Roles

Owner: The user who can set the password and read the password. Outsides: No one else should be able to set or read the password.

Executive Summary

Spent so much and found so much using these techniques.

Issues found

Severity	Number of Issues
High	2
Medium	0
Low	0
Informational	1
Total	3

Findings

High

[H-01] On-Chain Data Visibility Compromises Contract Confidentiality

Description: All data stored on the blockchain is publicly visible, which inherently precludes it from being confidential. The variable `PasswordStore::s_password`, intended as private, is expected to be accessed solely via the `PasswordStore::getPassword` function.

Impact: The visibility of on-chain data allows anyone to read the supposedly private passwords, thereby compromising the contract's confidentiality.

Proof of Concept: 1. Initiate a local blockchain using: `bash anvil` 2. Deploy the contract to the local chain with: `bash make deploy` 3. Apply the storage inspection tool (`cast storage`) to the

address of the deployed contract: `bash cast storage $CONTRACT_ADDRESS 1 --rpc-url http://127.0.0.1:8545` 4. Decode the hex storage output to a string with `cast parse-bytes32-string`: `bash cast parse-bytes32-string $OUTPUT_OF_THE_LAST_COMMAND`

Recommended Mitigation: A re-evaluation of the entire architecture is necessary. A potential solution could be to store an encrypted version of the password off-chain and retain only the encrypted form on-chain.

[H-02] Lack of Access Control in `PasswordStore::setPassword` Function

Description: The `PasswordStore::setPassword` function is currently unprotected, lacking checks on `msg.sender`, thereby permitting any user to exploit the function and alter the password. Project documentation stipulates that only the original user who stored the password should be able to invoke this password modification feature.

```
function setPassword(string memory newPassword) external {
    // @audit ACCESS CONTROL MISSING
    s_password = newPassword;
    emit SetNetPassword();
}
```

Impact: This vulnerability enables any user to change the contract-stored password, undermining the project's core principle.

Proof of Concept: Enhance the test suite by adding:

```
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(expectedPassword, actualPassword);
}
```

Recommended Mitigation: Implement access control within `PasswordStore::setPassword` as follows:

```
require(msg.sender == owner);
```

Informational

[I-01] Redundant NatSpec `@param` in `PasswordStore::getPassword`

Description: The NatSpec comments for `PasswordStore::getPassword` erroneously include a `@param` tag, which is unnecessary since the function does not accept any arguments.

Impact: Incorrect documentation may lead to confusion among users or developers interfacing with this contract.

```
/**
 * @notice This function allows only the owner to retrieve the password.
 * @info There are no input parameters for this function.
 * @param newPassword The new password to set. // INCORRECT
 */
function getPassword() external view returns (string memory) {
    if (msg.sender != s_owner) {
        revert PasswordStore__NotOwner();
    }
    return s_password;
}
```

Recommended Mitigation: Remove the irrelevant NatSpec `@param` line.

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

Gas