PasswordStore Audit Raport

Version 1.0

0xBlockPay

September 25, 2024

PasswordStore Audit Report

0xBlockPay

September 25, 2024

Prepared by: 0xBlockPay Lead Scurity Researcher: - eth0x

Table of Contents

Protocol Summary

Protocol PasswordStore allows the owner to store and retrieve a private password. Only the owner should be able to set access this passowrd. The protocol is designed only for one user.

Disclaimer

The 0xBlockPay 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

The findings describe in this document correspond the following repository and commit hash:

```
https://github.com/Cyfrin/3-passwordstore-audit
```

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

```
./src/
|__ PasswordStore.sol
```

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

Roles

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

Executive Summary

*We spend 10 hours with 1 auditor using foundry, aderyn and slither tools.

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 everyone and no longer private

Description: All data stored on-chain is visible to anyone, and can be readdirectly from the blockchain. The `PasswordStore::s_password` variable is intended to be a private variable and only accessed through the `PasswordStore::getPassword` function, which is intended to be only called by the owner of the contract.

Impact: Any one can read the private password,severly braking the functionality of the protocol.

Proof of Concept:

The below testcases hows how anyone can read the password directly from the blockchain.

Code

1. Create a locally running chain

```
make anvil
```

2. Deploy the contract to the chain

```
make deploy
```

3. Run the storage tool Use 1 because that's the storage slot ofs_password in the contract.

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

Output should be:

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

You can then parse this 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

[H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change password

Description: The function is set to be an `external` function however, the natspec of the function and overall purpose of the smart contarct is that `This function allows only the owner to set a new password.`

```
function setPassword(string memory newPassword) external {
  s_password = newPassword;
  emit SetNetPassword();
}
```

Impact: Any one can set/change the password of the contract, severly breaking the contract intended functionality.

Proof ofConcept:

Code Add the following to the PasswordStore.t.sol test file.

```
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.sol::setPassword` function

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

Medium

Low

Informational

[I-1] The `PasswordStore::getPassword` natspec indicates a parameter that does not exist, causing the natspec to be incorrect.

Description:

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

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

The `PasswordStore::getPassword` function signature is `getPassword()` while the natspec says it should be `getPassword(string)`.

Impact: The natspec is incorrect

Recommended Mitigation: Remove the incorrect natspace line.

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