audit/report.md

Prepared by: Guy Lead Security Researcher:

- Guy zilberblum

Table of Contents

- Table of Contents
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
  - Scope
  - Roles
- Executive Summary
  - Issues found
- Findings
- High
  - [H-1] Storing the password no cahin makes it visable to anyone , and no longer private
  - [H-2] 'passwordStore:: setPassword' has no access controls , meaning a non owner could change the password
- Informational
- Gas

Protocol Summary

A smart contract applicatoin for storing a password. Users should be able to store a password and then retrieve it later. Others should not be able to access the password.

Disclaimer

Guy 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 Guy 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 described in this document correspond the following commit hash***

""

```
 7d55682ddc4301a7b13ae9413095feffd9924566
```

""

Scope

```
./src/
└── PasswordStore.sol
```

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

we spent x hours with z auditors using y tools we found this and that

Issues found

Severity	number of issuse found
High	2
Medium	0
low	0
Info	1
Total	3

Findings

High

[H-1] Storing the password no cahin makes it visable to anyone , and no longer private

1. convince the protocol this is an issue
2. how bad the issue is
3. how to fix the issue

Description: All data stored on chain is visible to anyone and can be read directly from the block chain , the ***PasswordStore::s_pasword*** in intended to be privet varible and only accessed through ***PasswordStore::getPassword*** function witch is intended to be only called by the owner

we show a method of reading the data off chain below

Impact: Any one can read the password wich breaking the porpuse of the protocol

Proof of Concept: The below tese case shows how anyone cam read the password directly from the blockchain

1. Create a locally running chain ''' bash make anvil '''

2. Deploy the contract to the chain ''' make deploy '''

3. Run the storage tool we use '1' because thats the storage slot of ***s_password*** in the contract ''' cast storage <ADDRESS_HERE> 1 --rpc-url

you will get the output that looks like this '0x6d7950617373776f726400000000000000000000000000000000000000000014 ' you can then parse that hex to a string with: ''' cast parse-bytes32-string

0x6d7950617373776f72640000000000000000000000000000000000000000000014 ''' and then get an output of: ''' myPassword '''

Recommended Mitigation: Due to this the overall architecture of the contact 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 password However you'd also likely want to remove the view function as you wouldn't want the user to accidentally send a transaction with the password that dectypts your password

This function allows only the owner to set a new password

[H-2] 'passwordStore:: setPassword' has no access controls , meaning a non owner could change the password

Description: The 'passwordStore:: setPassword' function set to allow only the owner to set a password "" This function allows only the owner to set a new password"" but the the function missing this only owner key witch will make any one to acces the function and set a password as they like

"""javascript

```
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 of Concept: Add the following to(view on details) 'PasswordStore.test.sol' test file

▶ Подробные сведения

Recommended Mitigation: Add ab access control conditional to the ***setPassword*** function .

"""javascript

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

""""

Informational

Description: """ javascript

```
/*
* @notice This allows only the owner to retrieve the password.
@>* @param newPassword The new password to set.
*/
//  @audit the @param is not exist in the function
function getPassword() external view returns (string memory)
```

""" The ***PasswordStore::getPassword*** function signature is ***getPassword()*** while the docs says it should be ***getPassword(string)***

Impact: The docs is incorrect

Recommended Mitigation: remove the incorrect docs line """diff

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

"""

Gas