



Protocol Audit Report

Version 1.0

0xSquilliam

December 24, 2023

Protocol Audit Report

0xSquilliam

December 23, 2023

Prepared by: 0xSquilliam Lead Auditors:

- XXXXXXXX

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 on-chain makes it viable to anyone and no longer private
 - * [H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the password
 - Informational
 - * [I-1] `PasswordStore::getPassword` natspec indicates a paramter that doesnt exist, causing the natspec to be incorrect

Protocol Summary

Protocol does X, Y, Z

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
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

Commit Hash

```
1 1234567CommitHash89
```

Scope

```
1 ./src/  
2 #-- PasswordStore.sol
```

Roles

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

Executive Summary

Add some notes about how the audit went, types of things you found, etc.

We spent X hours with Z auditors using Y tools. etc.

Issues found

Sevterity	Number of issues found
Highs	2
Medium	0
Low	0
Info	1
Total	3

Findings

High

[H-1] Storing the password on-chain makes it viable 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 only accessed through the `PasswordStore : : getPassword` function, which is intended to be only 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: (Proof of Code)

1. Create a locally running chain

2. Deploy the contract to the chain

3. Run the storage tool We use 1 because that's the storage slot of `s_password` in the contract

```
1 you'll get an output that looks like this:  
2 `0  
   x6d7950617373776f726400000000000000000000000000000000000000000000000014  
   `  
3  
4 you can then parse that hex into a string with this:  
5 `cast parse-bytes32-string 0  
   x6d7950617373776f726400000000000000000000000000000000000000000000000014  
   `  
6  
7 and you'll get an output of:  
8 ```  
9 myPassword  
10 ````
```

Likelihood & Impact: -Impact: HIGH -Likelihood: HIGH -Severity: HIGH

Description: The `PasswordStore : : setPassword` function is set to be an `external` function (so anyone can call it), however, the natspec of the function and overall purpose of the smart contract is that `This function allows only the owner to set a new password.`

```
1 function setPassword(string memory newPassword) external {
2   @> //@audit - There are no access controls
3     s_password = newPassword;
4     emit SetNetPassword();
5 }
```

Impact: Anyone can set/change the password of the contract, severely breaking the intention of the contract.

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

```
1 function test_anyone_can_set_password(address randomAddress) public {
2   vm.assume(randomAddress != owner);
3   vm.prank(randomAddress);
4   string memory expectedPassword = "myNewPassword";
5   passwordStore.setPassword(expectedPassword);
6
7   vm.prank(owner);
8   string memory actualPassword = passwordStore.getPassword();
9   assertEq(actualPassword, expectedPassword);
10 }
```

Recommended Mitigation: Add an access control conditional to the `setPassword` function.

```
1 if(msg.sender != s_owner){
2   revert PasswordStore_notOwner();
3 }
```

Likelihood & Impact: -Impact: HIGH -Likelihood: HIGH -Severity: HIGH

Informational

[I-1] PasswordStore::getPassword natspec indicates a paramter that doesnt exist, causing the natspec to be incorrect

Description:

```
1 /*
2  * @notice This allows only the owner to retrieve the password.
3  * @param newPassword The new password to set.
4  */
5 function getPassword() external view returns (string memory) {
```

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

Impact: The natspec is incorrect

Recommended Mitigation: Remove the incorrect natspec line.

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

Likelihood & Impact: -Impact: NONE -Likelihood: HIGH -Severity: INFORMATIONAL/GAS/Non-Crits