Protocol Audit Report

Version 1.0

bumble.c

July 5, 2024

Protocol Audit Report

bumble.c

July 2024

Prepared by: Bumble Lead Security Researcher:

- Bumble.c
- Protocol Summary
- Disclaimer
- Risk Classification
- Audit Details
  - Scope
  - Roles
- Executive Summary
  - Issues found
- Findings
  - High
    * [H-1] Storing the password on chain makes it visible for anyone, and no longer private
    * [H-2] `PasswordStore::setPassword` has no access controls, meaning a non-owner could change the password
  - Informational
    * [S-#] The `PasswordStore::getPassword` natspec indicates a parameter that doenst exist, causing the natspec to be incorrect

Protocol Summary

PasswordStore is a simple contract that allows a user to store a password on-chain. The password can be set by the owner of the contract, and retrieved by the owner of the contract.

Disclaimer

The Bumble 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 described in this document correspond the following commit hash.

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

- In Scope:

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

Roles

- Owner - The owner of the contract who can set and retrieve the password.
- Outsiders - Anyone who is not the owner of the contract.

Executive Summary

The PasswordStore contract is a simple contract that allows a user to store a password on-chain. The password can be set by the owner of the contract, and retrieved by the owner of the contract. The findings were found in the `PasswordStore.sol` contract. We spent 4 hours auditing the contract, with 2 auditors using the CodeHawks methodology and 4 tools.

Issues found

Severity	Numbers 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 for 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 any data off chain below.

Impact: Anyone can read the private password, severely breaking the functionality of the protocol.

Proof of Concept: (Proof of Code)

The below test case shows how anyone can read the password directly from the blockchain.

1. Start a local node

```
make anvil
```

2. Deploy

This will default to your local node. You need to have it running in another terminal in order for it to deploy.

```
make deploy
```

3. Run the storage tool

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

```
cast storage <ADDRESS_HERE> 1 --rpc-url http://localhost:8545
```

You will get an output that looks like:

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

You can then parse that 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 re-evaluated. 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 decrpyt the password. However, you'd also likely want to remove the view function as you wouldnt want the user to accidentally send a transaction with the password that decrypts it.

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

Description: The `PasswordStore::setPassword` function is set to be an external function, 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
2     function setPassword(string memory newPassword) external {
3 @>      // @audit - There are no access controls
4         s_password = newPassword;
5         emit SetNetPassword();
6     }
```

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

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

Click to see the code

```
1     function test_anyone_can_set_password(address randomAddress) public
          {
2         vm.prank(randomAddress);
3         passwordStore.setPassword("TestPassword");
4
5         vm.prank(address(msg.sender));
6         string memory password = passwordStore.getPassword();
7         assertEq(password, "TestPassword");
8     }
```

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

```
1 if (msg.sender != owner) {
2   revert("Only the owner can set the password");
3 }
```

Informational

[S-#] The `PasswordStore::getPassword` natspec indicates a parameter that doenst 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 say it should be `getPassword(string)`.

Impact: The natspec is incorrect.

Recommended Mitigation: Remove the incorrect natspec line.

```
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) {
```