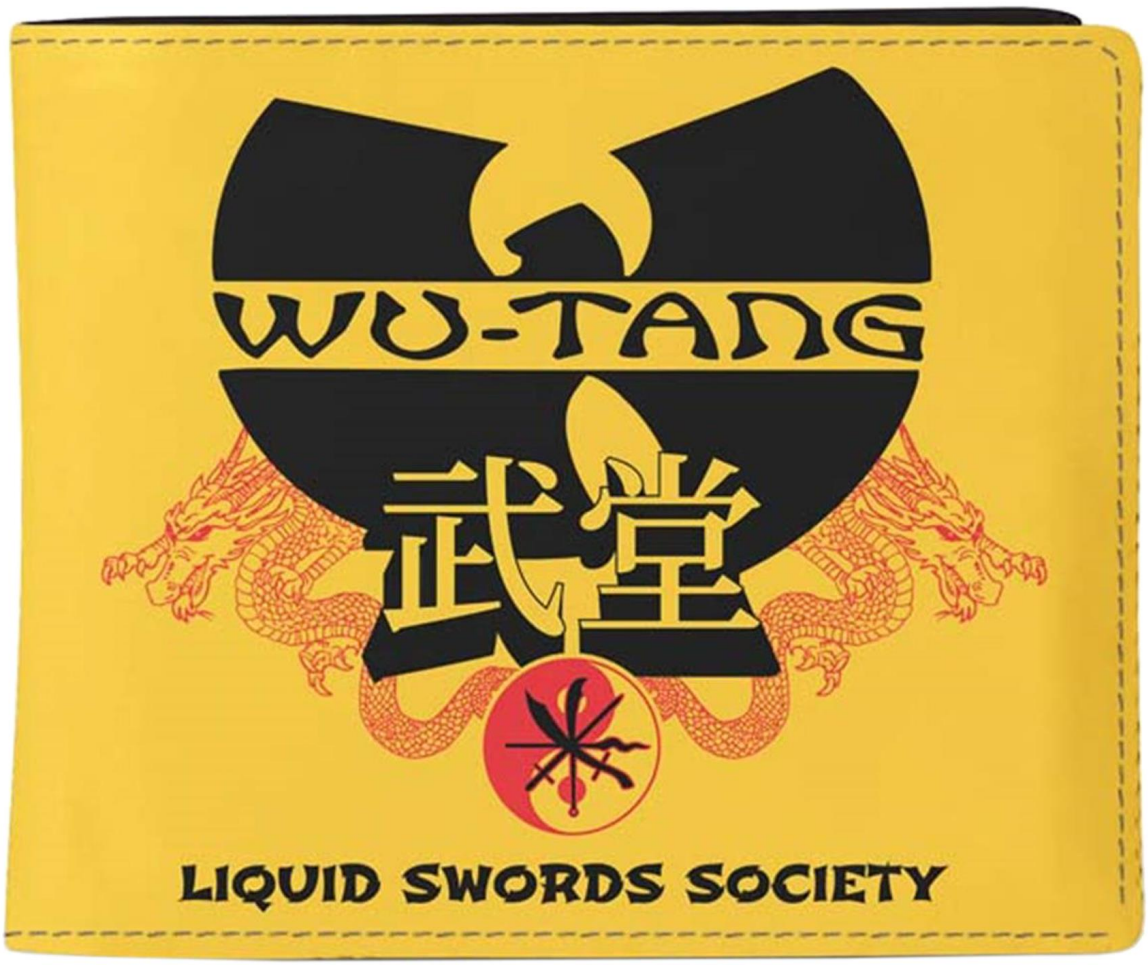

Protocol Audit Report

Prepared by: Georgi Penchev

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 visible to anyone, and no longer private.
    - [H-2] TITLE `PasswordStore::setPassword` has no access control, meaning a non-owner could change the password.
  - Informational
    - [I-1] TITLE The `PasswordStore::getPassword` natspec indicates a parameter that does not exist, causing the natspec to be incorrect

Protocol Summary

PasswordStore is a protocol dedicated to store and retrieve of user's password. The protocol is designed to be used by a single user, and not designed to be used by multiple users. Only the owner should be able to set and access the password.

Disclaimer

The Georgi Penchev 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 the document correspond to the following commit has:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

```
./src/
└── PasswordStore.sol
```

Roles

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

Executive Summary

I spent hours to summarise this content with 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 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 varibale and only accessed through the `PasswordStore::getPassword` function, which is intended to be 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, severity 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.

make anvil

cast storage 0x5FbDB2315678afecb367f032d93F642f64180aa3 1 --rpc-url http://127.0.0.1:8545

cast parse-bytes32-string 0x6d7950617373776f726400000000000000000000000000000000000000000014

Recommended Mitigation: We could encrypt the password off-chain and then store the encypted password on-chain, however this would require the user to remember password off chain to decrypt. However we could remove the view function as you dont want the user to accidently send a transaction with the password that decrypts your password

[H-2] TITLE `PasswordStore::setPassword` has no access control, 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 can only be set by the owner`

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

Proof of Concept: 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(expectedPassword, actualPassword);
    }
```

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

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

Informational

[I-1] TITLE 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 natspec line.

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